AsyncDex

Release 0.4

PythonCoderAS

May 17, 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 and print the authors of the manga:

```
manga = await client.random_manga()
print(f"{manga.id}: {manga.titles.en.primary}")
await manga.load_authors()
print(f"Author of {manga.titles.en.primary}: {manga.authors[0].name}")
```

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()
response.close()
[print(item["data"]["id"]) for item in json["results"]]
```

CHAPTER

TWO

CHANGELOG

2.1 v0.4

2.1.1 Added

- *return_date_string()*
- *download_all()*
- *Pager.limit* to limit total responses,
- *Pager.as_list()*
- *Tag.descriptions*
- *Tag.group*
- *TagDict*
- Allow the creation of *User* objects if the ID is in the base data dictionary.
- *Demographic.NONE*
- *OrderDirection*
- *TagMode*
- *AuthorListOrder*
- *ChapterListOrder*
- *GroupListOrder*
- *MangaListOrder*
- **Methods added to *MangadexClient*:**
  - *get_groups()*
  - *get_chapters()*
  - *get_authors()*
  - *get_mangas()*
  - *report_page()*
  - *MangadexClient.close()*

2.1.2 Changed

- Changed *download_chapter()* so that directories are not created until all pages are retrieved.
- Moved *Chapter.get_page()* to *MangadexClient.get_page()*.

2.1.3 Fixed

- Fixed *Pager.__anext__()* so it does not need to complete all requests before returning the first batch of statements. This will drastically improve performance if all items aren't needed immediately (such as making further requests with returned data).
- Fixed a bug where the chapter list would clear itself when filtered.
- Fixed a bug where *download_chapter()* would not try again due to certain errors such as establishing a connection.
- Fixed *Chapter.pages()* so it respects the `forcePort443` parameter.

2.2 v0.3

2.2.1 Added

- Added a ratelimit on the */at-home/server/{id}* path to match the 5.0.2 release of the MD API.
- Added a global ratelimit for 5 req/s to match the ratelimit set by the MD API.
- *DuplicateResolutionAlgorithm*
- *Chapter*
- *ChapterList*
- *Group*
- *Manga.chapters*
- *Pager*
- *User*
- **Methods added to *MangadexClient*:**
  - *get_chapter()*
  - *batch_chapters()*
  - *get_user()*
  - *logged_in_user()*
  - *ping()*
  - *convert_legacy()*
  - *get_group()*
  - *batch_groups()*
- *AttrDict.first()* and `DefaultAttrDict.first()`
- *Interval*

- *InclusionExclusionPair*

2.2.2 Changed

- *Manga.last_volume* and *Manga.last_chapter* both are now Strings.
- Made all of the `batch_*` methods on the Client class parallel. This will speed up batch requests over the size of 100 items fivefold.

2.2.3 Fixed

- *Manga.last_chapter* did not account for floating point variables.
- Changed *Model.__repr__()* to properly show the delimiters for strings.
- *MangadexClient.__aexit__()* will now close the underlying session object.
- Fixed a bug in `Client.request()` that prevented the use of non-string and non-iterable objects such as integers and floats.
- Added a client-side fix for the incorrect spelling of the word `hiatus` on the MangaDex API.
- Fixed a typo on *Demographic.JOSEI* where the term "josei" was actually spelled "josel".
- Added a message to *Unauthorized*.
- Fixed a bunch of places where requests are not properly closed.
- Changed the value of `MangaStatus.ABANDONED` to match new API specifications.
- Fixed a bug in the retry mechanism of `Client.request()` that added the parameters for a second time.

2.3 v0.2

2.3.1 Added

- **The 6 enums:**
    1. *Demographic*
    2. *MangaStatus*
    3. *FollowStatus*
    4. *ContentRating*
    5. *Visibility*
    6. *Relationship*
- *Missing*
- *InvalidID*
- **Models:**
    - *Model*
    - *Manga*
    - *Tag*

    - *Author*
- *tag_cache* inside of *MangadexClient*
- **Methods to *MangadexClient*:**
    - *refresh_tag_cache()*
    - *get_tag()*
    - *get_manga()*
    - *random_manga()*
    - *batch_authors()*
    - *get_author()*
    - *batch_mangas()*
- *DatetimeMixin*
- *TitleList*
- *AttrDict*
- *DefaultAttrDict*
- *copy_key_to_attribute()*
- *parse_relationships()*

2.4 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.

> **Warning:** The client cannot ratelimit effectively if multiple clients are running on the same program. Furthermore, the ratelimit may not work if multiple other people are accessing the MangaDex API at the same time or the client is running on a shared network.

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.
- **api_url** (*str*) – The base URL for the MangaDex API. Useful for private instances or a testing environment. Should not include a trailing slash.
- **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`.

async __aenter__()
Allow the client to be used with `async with` syntax similar to `aiohttp.ClientSession`.

async __aexit__(*exc_type: Optional[Type[BaseException]]*, *exc_val: Optional[BaseException]*, *exc_tb: Optional[traceback]*)
Exit the client. This will also close the underlying session object.

__repr__() → str
Provide a string representation of the client.

Returns The string representation

Return type str

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 batch_authors(**authors: asyncdex.models.author.Author*)
Updates a lot of authors at once, reducing the time needed to update tens or hundreds of authors.

New in version 0.2.

Parameters **authors** (*Tuple[Author, ...]*) – A tuple of all the authors (and artists) to update.

async batch_chapters(**chapters: asyncdex.models.chapter.Chapter*)
Updates a lot of chapters at once, reducing the time needed to update tens or hundreds of chapters.

New in version 0.3.

See also:

ChapterList.get().

Parameters **chapters** (*Tuple[Chapter, ...]*) – A tuple of all the chapters to update.

async batch_groups(**groups: asyncdex.models.group.Group*)
Updates a lot of groups at once, reducing the time needed to update tens or hundreds of groups.

New in version 0.3.

Parameters **groups** (*Tuple[Group, ...]*) – A tuple of all the groups to update.

async batch_mangas(**mangas: asyncdex.models.manga.Manga*)
Updates a lot of mangas at once, reducing the time needed to update tens or hundreds of mangas.

New in version 0.2.

Parameters **mangas** (*Tuple[Manga, ...]*) – A tuple of all the mangas to update.

async close()
Close the client.

New in version 0.4.

async convert_legacy(*model: Type[_LegacyModelT]*, *ids: List[int]*) → Dict[int, _LegacyModelT]
Convert a list of legacy IDs to the new UUID system.

New in version 0.3.

Parameters

- **model** (*Type[Manga, Chapter, Tag, Group]*) – The model that represents the type of conversion. The endpoint allows conversions of old mangas, chapters, tags, and groups.
- **ids** (*List[int]*) – The list of integer IDs to convert.

Returns

A dictionary mapping old IDs to instances of the model with the new UUIDs.

Note: Except for tags, all other models will be lazy models. However, batch methods exist for all other models.

Return type Dict[int, *Model*]

get_author(*id: str*) → asyncdex.models.author.Author
Get an author using it's ID.

New in version 0.2.

Note: This method can also be used to get artists, since they are the same class.

Warning: This method returns a **lazy** Author instance. Call *Author.fetch()* on the returned object to see any values.

Parameters **id** (*str*) – The author's UUID.

Returns A *Author* object.

Return type *Author*

get_authors(*, *name: Optional[str] = None*, *order: Optional[asyncdex.list_orders.AuthorListOrder] = None*, *limit: Optional[int] = None*) → *asyncdex.models.pager.Pager*[asyncdex.models.author.Author]
Creates a *Pager* for authors.

New in version 0.4.

Usage:

```
async for author in client.get_authors(name="Author Name"):
    ...
```

Parameters

- **name** (*str*) – The name to search for.
- **order** (AuthorListOrder) – The order to sort the authors[1].
- **limit** (*int*) – Only return up to this many authors.

Note: Not setting a limit when you are only interested in a certain amount of responses may result in the Pager making more requests than necessary, consuming ratelimits.

Returns A Pager for the authors.

Return type *Pager*

get_chapter(*id: str*) → asyncdex.models.chapter.Chapter

Get a chapter using it's ID.

New in version 0.3.

See also:

ChapterList.get().

Warning: This method returns a **lazy** Chapter instance. Call *Chapter.fetch()* on the returned object to see any values.

Parameters **id** (*str*) – The chapter's UUID.

Returns A *Chapter* object.

Return type *Chapter*

get_chapters(**, title: Optional[str] = None, groups: Optional[Sequence[Union[str, asyncdex.models.group.Group]]] = None, uploader: Optional[Union[str, asyncdex.models.user.User]] = None, manga: Optional[Union[str, asyncdex.models.manga.Manga]] = None, volume: Optional[str] = None, chapter_number: Optional[str] = None, language: Optional[str] = None, created_after: Optional[datetime.datetime] = None, updated_after: Optional[datetime.datetime] = None, published_after: Optional[datetime.datetime] = None, order: Optional[*asyncdex.list_orders.ChapterListOrder*] = None, limit: Optional[int] = None*) → *asyncdex.models.pager.Pager*[asyncdex.models.chapter.Chapter]

Gets a *Pager* of chapters.

New in version 0.4.

Usage:

```
async for chapter in client.get_chapters(chapter_number="1"):
    ...
```

Parameters

- **title** (*str*) – The title of the chapter.

[1] This parameter is undocumented in the API as of May 16, 2021. The inclusion of this parameter can be found in the changelog of the v5.0.6 release of the API, found in the MangaDex Discord.

- **groups** (*List[Union[str, Group]]*) – Chapters made by one of the groups in the given list. A group can either be the UUID of the group in string format or an instance of *Group*.
- **uploader** (*Union[str, User]*) – The user who uploaded the chapter. A user can either be the UUID of the user in string format or an instance of *User*.
- **manga** (*Union[str, Manga]*) – Chapters that belong to the given manga. A manga can either be the UUID of the manga in string format or an instance of *Manga*.

Note: If fetching all chapters for one manga, it is more efficient to use *ChapterList.get()* instead.

- **volume** (*str*) – The volume that the chapter belongs to.
- **chapter_number** (*str*) – The number of the chapter.
- **language** (*str*) – The language of the chapter.
- **created_after** (*datetime*) – Get chapters created after this date.

Note: The datetime object needs to be in UTC time. It does not matter if the datetime if naive or timezone aware.

- **updated_after** (*datetime*) – Get chapters updated after this date.

Note: The datetime object needs to be in UTC time. It does not matter if the datetime if naive or timezone aware.

- **published_after** (*datetime*) – Get chapters published after this date.

Note: The datetime object needs to be in UTC time. It does not matter if the datetime if naive or timezone aware.

- **order** (ChapterListOrder) – The order to sort the chapters.
- **limit** (*int*) – Only return up to this many chapters.

Note: Not setting a limit when you are only interested in a certain amount of responses may result in the Pager making more requests than necessary, consuming ratelimits.

Returns A Pager for the chapters.

Return type *Pager*

get_group(*id: str*) → asyncdex.models.group.Group

Get a group using it's ID.

New in version 0.3.

> **Warning:** This method returns a **lazy** Group instance. Call `Group.fetch()` on the returned object to see any values.

Parameters **id** (*str*) – The group's UUID.

Returns A *Group* object.

Return type *Group*

get_groups(*, *name: Optional[str] = None*, *order: Optional[*asyncdex.list_orders.GroupListOrder*] = None*, *limit: Optional[int] = None*) → *asyncdex.models.pager.Pager*[asyncdex.models.group.Group]

Creates a *Pager* for groups.

New in version 0.4.

Usage:

```
async for group in client.get_groups(name="Group Name"):
    ...
```

Parameters

- **name** (*str*) – The name to search for.
- **order** (*GroupListOrder*) – The order to sort the groups[?].
- **limit** (*int*) – Only return up to this many groups.

> **Note:** Not setting a limit when you are only interested in a certain amount of responses may result in the Pager making more requests than necessary, consuming ratelimits.

Returns A Pager for the groups.

Return type *Pager*

get_manga(*id: str*) → asyncdex.models.manga.Manga

Get a manga using it's ID.

New in version 0.2.

See also:

search().

> **Warning:** This method returns a **lazy** Manga instance. Call `Manga.fetch()` on the returned object to see any values.

Parameters **id** (*str*) – The manga's UUID.

Returns A *Manga* object.

Return type *Manga*

get_mangas(*, *title: Optional[str] = None*, *authors: Optional[List[Union[str, asyncdex.models.author.Author]]] = None*, *artists: Optional[List[Union[str, asyncdex.models.author.Author]]] = None*, *year: Optional[int] = None*, *included_tags: Optional[List[Union[str, asyncdex.models.tag.Tag]]] = None*, *included_tag_mode: asyncdex.enum.TagMode = <TagMode.AND: 'AND'>*, *excluded_tags: Optional[List[Union[str, asyncdex.models.tag.Tag]]] = None*, *excluded_tag_mode: asyncdex.enum.TagMode = <TagMode.OR: 'OR'>*, *status: Optional[List[asyncdex.enum.MangaStatus]] = None*, *languages: Optional[List[str]] = None*, *demographic: Optional[List[asyncdex.enum.Demographic]] = None*, *rating: Optional[List[asyncdex.enum.ContentRating]] = None*, *created_after: Optional[datetime.datetime] = None*, *updated_after: Optional[datetime.datetime] = None*, *order: Optional[asyncdex.list_orders.MangaListOrder] = None*, *limit: Optional[int] = None*) → *asyncdex.models.pager.Pager*[asyncdex.models.manga.Manga]

Gets a *Pager* of mangas.

New in version 0.4.

Usage:

```
async for manga in client.search(title="Solo Leveling"):
    ...
```

Parameters

- **title** (*str*) – The title of the manga.
- **authors** (*List[Union[str, Author]]*) – Mangas made by the given authors. An author may be represented by a string containing their UUID or an instance of *Author*.
- **artists** (*List[Union[str, Author]]*) – Mangas made by the given artists. An artist may be represented by a string containing their UUID or an instance of *Author*.
- **year** (*int*) – The year the manga was published.
- **included_tags** (*List[Union[str, Tag]]*) – A list of tags that should be present. A tag may be represented by a string containing the tag's UUID or an instance of *Tag*.
- **included_tag_mode** (TagMode) – The mode to use for the included tags. Defaults to *TagMode.AND*.
- **excluded_tags** (*List[Union[str, Tag]]*) – A list of tags that should not be present. A tag may be represented by a string containing the tag's UUID or an instance of *Tag*.
- **excluded_tag_mode** (TagMode) – The mode to use for the excluded tags. Defaults to *TagMode.OR*.
- **status** (*List[MangaStatus]*) – A list of *MangaStatus*es representing possible statuses.
- **languages** (*List[str]*) – A list of language coes.
- **demographic** (*List[Demographic]*) – A list of *Demographic*s representing possible demographics.
- **rating** – A list of *ContentRating*s representing possible content ratings.
- **created_after** (*datetime*) – Get mangas created after this date.

Note: The datetime object needs to be in UTC time. It does not matter if the datetime if naive or timezone aware.

- **updated_after** (*datetime*) – Get mangas updated after this date.

Note: The datetime object needs to be in UTC time. It does not matter if the datetime if naive or timezone aware.

- **limit** (*int*) – Only return up to this many mangas.

Note: Not setting a limit when you are only interested in a certain amount of responses may result in the Pager making more requests than necessary, consuming ratelimits.

- **order** (MangaListOrder) – The order to sort the mangas.

Returns A Pager with the manga entries.

Return type *Pager*

async get_page(*url: str*) → aiohttp.client_reqrep.ClientResponse

A method to download one page of a chapter, using the URLs from *pages()*. This method is more low-level so that it is not necessary to download all pages at once. This method also respects the API rules on downloading pages.

Parameters **url** (*str*) – The URL to download.

Raises aiohttp.ClientResponseError if a 4xx or 5xx response code is returned.

Returns The aiohttp.ClientResponse object containing the image.

Return type aiohttp.ClientResponse

async get_session_token()

Get the session token and store it inside the client.

async get_tag(*id: str*) → asyncdex.models.tag.Tag

Get a tag using it's ID.

New in version 0.2.

Finding a Tag by Name

Finding a tag by name is a feature that many people want. However, there is no endpoint that exists in the API that lets us provide a name and get back a list of Tags that match the name. It is not needed, as there only exists a relatively amount of tags, which can be loaded from a single request.

The client maintains a cache of the tags in order to lower memory usage and allow tag updates to be easily distributed to all mangas, since there are a relatively small amount of tags compared to authors, chapters, mangas, and users. The client also provides a method to completely load the tag list and update the tag cache, *refresh_tag_cache()*. The tag cache is stored in *tag_cache*, Using this property, it is possible to iterate over the tag list and preform a simple name matching search to find the tag(s) that you want. An example implementation of a tag search method is provided as such:

```
from asyncdex import MangadexClient, Tag
from typing import List

def search_tags(client: MangadexClient, phrase: str) -> List[Tag]:
    phrase = phrase.replace(" ", "") # Remove spaces so "sliceoflife" and
↪"slice of life" match.
    results: List[Tag] = []
    for tag in client.tag_cache:
        for name in tag.names.values():
            if phrase in name.replace(" ", "").lower():
                results.append(tag)
                break
    return results
```

Parameters **id** (*str*) – The tag's UUID.

Returns A *Tag* object.

Return type *Tag*

get_user(*id: str*) → asyncdex.models.user.User
Get a user using it's ID.

New in version 0.3.

> **Warning:** This method returns a **lazy** User instance. Call *User.fetch()* on the returned object to see any values.

Parameters **id** (*str*) – The user's UUID.

Returns A *User* object.

Return type *User*

async logged_in_user() → asyncdex.models.user.User
Get the user that is currently logged in.

New in version 0.3.

Returns A *User* object.

Return type *User*

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.

async ping()
Ping the server. This will throw an error if there is any error in making connections, whether with the client or the server.

New in version 0.3.

async random_manga() → asyncdex.models.manga.Manga
Get a random manga.

New in version 0.2.

Returns A random manga.

Return type *Manga*

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

async refresh_tag_cache()
Refresh the internal tag cache.

New in version 0.2.

See also:

tag_cache

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 report_page(*url: str*, *success: bool*, *response_length: int*, *duration: int*, *cached: bool*)
Report a page to the MangaDex@Home network.

New in version 0.4.

See also:

`Client.get_page()`, which will automatically call this method for you.

Parameters

- **url** (*str*) – The URL of the image.
- **success** (*bool*) – Whether or not the URL was successfully retrieved.
- **response_length** (*int*) – The length of the response, whether or not it was a success.
- **duration** (*int*) – The time it took for the request, including downloading the content if it existed, **in milliseconds**.
- **cached** (*bool*) – Whether or not the request was cached (The `X-Cache` header starting with the value `HIT`).

async request(*method: str*, *url: str*, ***, *params: Optional[Mapping[str, Optional[Union[str, Sequence[str], bool, float]]]] = 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.

Note: The request method will log all URLs that are requested. Enable logging on the `asyncdex` logger to view them. These requests are made under the `INFO` level. Retries are also logged on the `WARNING` level.

Changed in version 0.3: Added a global (shared between all requests made in the client) ratelimit.

Changed in version 0.4: Added better handling of string items.

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

search(*, *title: Optional[str] = None*, *authors: Optional[List[Union[str, asyncdex.models.author.Author]]] = None*, *artists: Optional[List[Union[str, asyncdex.models.author.Author]]] = None*, *year: Optional[int] = None*, *included_tags: Optional[List[Union[str, asyncdex.models.tag.Tag]]] = None*, *included_tag_mode: asyncdex.enum.TagMode = <TagMode.AND: 'AND'>*, *excluded_tags: Optional[List[Union[str, asyncdex.models.tag.Tag]]] = None*, *excluded_tag_mode: asyncdex.enum.TagMode = <TagMode.OR: 'OR'>*, *status: Optional[List[asyncdex.enum.MangaStatus]] = None*, *languages: Optional[List[str]] = None*, *demographic: Optional[List[asyncdex.enum.Demographic]] = None*, *rating: Optional[List[asyncdex.enum.ContentRating]] = None*, *created_after: Optional[datetime.datetime] = None*, *updated_after: Optional[datetime.datetime] = None*, *order: Optional[asyncdex.list_orders.MangaListOrder] = None*, *limit: Optional[int] = None*) → *asyncdex.models.pager.Pager*[asyncdex.models.manga.Manga]

Alias for *get_mangas()*.

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.

tag_cache: *asyncdex.models.tag.TagDict*
A cache of tags. This cache will be used to lower the amount of tag objects, and allows for easily updating the attributes of tags. This cache can be refreshed manually by either calling *refresh_tag_cache()* or fetching data for any tag object.

New in version 0.2.

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

class asyncdex.models.abc.**Model**(*client: MangadexClient*, ***, *id: Optional[str] = None*, *version: int = 0*, *data: Optional[Dict[str, Any]] = None*)
An abstract model. Cannot be instantiated.

New in version 0.2.

Raises *Missing* if there is no valid ID in the model after parsing provided data.

Parameters **data** (*Dict[str, Any]*) – The data received from the server. May be None if there is no data yet.

__eq__(*other: _T*) → bool
Check if two models are equal to each other.

Parameters **other** (Model) – Another model. Should be the same type as the model being compared.

Returns Whether or not the models are equal.

Return type bool

__hash__()
Return hash(self).

__ne__(*other: _T*) → bool
Check if two models are not equal to each other.

Parameters **other** (Model) – Another model. Should be the same type as the model being compared.

Returns Whether or not the models are equal.

Return type bool

__repr__() → str
Returns a string version of the model useful for development.

__str__() → str
Returns a string representation of the model, usually it's id.

client: MangadexClient
The client that created this model.

abstract async fetch()
Fetch the data to complete any missing non-critical values.

Raises *InvalidID* if an object with the ID does not exist.

id: str
A UUID that represents this item.

abstract parse(*data: Dict[str, Any]*)
Parse the data received from the server.

Parameters **data** (*Dict[str, Any]*) – The data from the server.

transfer(*new_obj: _T*)
Transfer data from a new object to the current object.

Parameters **new_obj** (Model) – The new object. Should be the same type as the current model.

version: int
The version of the model.

class asyncdex.models.**Author**(*client: MangadexClient*, *, *id: Optional[str] = None*, *version: int = 0*, *data: Optional[Dict[str, Any]] = None*)
A *Model* representing an individual author.

Note: Artists and authors are stored identically and share all properties.

New in version 0.2.

__eq__(*other: _T*) → bool
Check if two models are equal to each other.

Parameters **other** (Model) – Another model. Should be the same type as the model being compared.

Returns Whether or not the models are equal.

Return type bool

__hash__()
Return hash(self).

__ne__(*other: _T*) → bool
Check if two models are not equal to each other.

Parameters **other** (Model) – Another model. Should be the same type as the model being compared.

Returns Whether or not the models are equal.

Return type bool

__str__() → str
Returns a string representation of the model, usually it's id.

biographies: *asyncdex.utils.DefaultAttrDict*[Optional[str]]
A *DefaultAttrDict* holding the biographies of the author.

client: MangadexClient
The client that created this model.

created_at: datetime
A datetime.datetime representing the object's creation time.

See also:

modified_at()

Note: The datetime is **timezone aware** as it is parsed from an ISO-8601 string.

---

async fetch()
Fetch the data to complete any missing non-critical values.

Raises *InvalidID* if an object with the ID does not exist.

id: str
A UUID that represents this item.

image: Optional[str]
An image of the author, if available.

async load_mangas()
Shortcut method that calls Client.batch_mangas() with the mangas that belong to the author.

Roughly equivalent to:

```
await client.batch_mangas(*author.mangas)
```

mangas: List[*Manga*]
A list of all the mangas that the author has written.

Note: In order to efficiently get all mangas in one go, use:

```
await client.batch_mangas(*author.mangas)
```

---

property modified_at
The last time the object was modified. This will return the creation time if the object was never updated after creation, or the modification time if it has.

See also:

created_at, *updated_at*

Returns

The last time the object was changed as a datetime.datetime object.

Note: The datetime is **timezone aware** as it is parsed from an ISO-8601 string.

Return type datetime.datetime

name: str
The name of the author.

parse(*data: Dict[str, Any]*)
Parse the data received from the server.

Parameters **data** (*Dict[str, Any]*) – The data from the server.

transfer(*new_obj: _T*)
Transfer data from a new object to the current object.

Parameters **new_obj** (Model) – The new object. Should be the same type as the current model.

updated_at: Optional[datetime]
A datetime.datetime representing the last time the object was updated. May be None if the object was never updated after creation.

See also:

modified_at()

Note: The datetime is **timezone aware** as it is parsed from an ISO-8601 string.

---

version: int
The version of the model.

class asyncdex.models.**Chapter**(*client: MangadexClient*, *, *id: Optional[str] = None*, *version: int = 0*, *data: Optional[Dict[str, Any]] = None*)
A *Model* representing an individual chapter.

New in version 0.3.

__eq__(*other: _T*) → bool
Check if two models are equal to each other.

Parameters **other** (Model) – Another model. Should be the same type as the model being compared.

Returns Whether or not the models are equal.

Return type bool

__ge__(*other: _T*) → bool
Compares the two object's creation times to find if the current model's creation time is greater than or equal to the other model's creation time.

New in version 0.3.

Parameters **other** (DatetimeMixin) – The other model.

Returns Whether or not the current model's creation time is greater than or equal to the other model's creation time.

Return type bool

__gt__(*other: _T*) → bool
Compares the two object's creation times to find if the current model's creation time is greater than the other model's creation time.

New in version 0.3.

Parameters **other** (DatetimeMixin) – The other model.

Returns Whether or not the current model's creation time is greater than the other model's creation time.

Return type bool

__hash__()
Return hash(self).

__le__(*other: _T*) → bool
Compares the two object's creation times to find if the current model's creation time is less than or equal to the other model's creation time.

New in version 0.3.

Parameters **other** (DatetimeMixin) – The other model.

Returns Whether or not the current model's creation time is less than or equal to the other model's creation time.

Return type bool

__lt__(*other: _T*) → bool
Compares the two object's creation times to find if the current model's creation time is less than the other model's creation time.

New in version 0.3.

Parameters **other** (DatetimeMixin) – The other model.

Returns Whether or not the current model's creation time is less than the other model's creation time.

Return type bool

__ne__(*other: _T*) → bool
Check if two models are not equal to each other.

Parameters **other** (Model) – Another model. Should be the same type as the model being compared.

Returns Whether or not the models are equal.

Return type bool

__str__() → str
Returns a string representation of the model, usually it's id.

client: MangadexClient
The client that created this model.

created_at: datetime
A datetime.datetime representing the object's creation time.

See also:

modified_at()

Note: The datetime is **timezone aware** as it is parsed from an ISO-8601 string.

---

data_saver_page_names: List[str]
A list of strings containing the filenames of the data saver pages.

See also:

page_names

async download_chapter(*, *folder_format: str = '{manga}/{chapter_num}{separator}{title}'*, *file_format: str = '{num}'*, *as_bytes_list: bool = False*, *overwrite: bool = True*, *retries: int = 3*, *use_data_saver: bool = False*, *ssl_only: bool = False*) → Optional[List[bytes]]

Download all of the pages of the chapter and either save them locally to the filesystem or return the raw bytes.

Parameters

- **folder_format** (*str*) – The format of the folder to create for the chapter. The folder can already be existing. The default format is `{manga}/{chapter_num}{separator}{chapter_title}`.

  Note: Specify `.` if you want to save the pages in the current folder.

  Available variables:

  - `{manga}`: The name of the manga. If the chapter's manga object does not contain a title object, it will be fetched.
  - `{chapter_num}`: The number of the chapter, if it exists.
  - `{separator}`: A separator if both the chapter's number and title exists.
  - `{title}`: The title of the chapter, if it exists.

- **file_format** (*str*) – The format of the individual image file names. The default format is `{num}`.

  Note: The file extension is applied automatically from the real file name. There is no need to include it.

  Available variables:

  - `{num}`: The numbering of the image files starting from 1. This respects the order the images are in inside of *page_names*.
  - `{num0}`: The same as `{num}` but starting from 0.
  - `{name}`: The actual filename of the image from *page_names*, without the file extension.

- **as_bytes_list** (*bool*) – Whether or not to return the pages as a list of raw bytes. Setting this parameter to `True` will ignore the value of the `folder_format` parameter.
- **overwrite** (*bool*) – Whether or not to override existing files with the same name as the page. Defaults to `True`.
- **retries** (*int*) – How many times to retry a chapter if a MD@H node does not let us download the pages. Defaults to `3`.
- **use_data_saver** (*bool*) – Whether or not to use the data saver pages or the normal pages. Defaults to `False`.
- **ssl_only** (*bool*) – Whether or not the given URL has port `443`. Useful if your firewall blocks outbound connections to ports that are not port `443`. Defaults to `False`.

  Note: This will lower the pool of available clients and can cause higher download times.

Raises aiohttp.ClientResponseError if there is an error after all retries are exhausted.

Returns A list of byte strings if as_bytes_list is True else None.

Return type Optional[List[bytes]]

async fetch()
Fetch the data to complete any missing non-critical values.

Raises *InvalidID* if an object with the ID does not exist.

async get_page(*url: str*)
Alias for *MangadexClient.get_page()*.

Deprecated since version 0.4.

groups: List[asyncdex.models.group.Group]
The groups that uploaded this chapter.

hash: str
The chapter's hash.

id: str
A UUID that represents this item.

language: str
The language of the chapter.

async load_groups()
Shortcut method that calls Client.batch_groups() with the groups that belong to the group.

Roughly equivalent to:

```
await client.batch_groups(*user.groups)
```

manga: *Manga*
The manga that this chapter belongs to.

property modified_at
The last time the object was modified. This will return the creation time if the object was never updated after creation, or the modification time if it has.

See also:

created_at, *updated_at*

Returns

The last time the object was changed as a datetime.datetime object.

Note: The datetime is **timezone aware** as it is parsed from an ISO-8601 string.

Return type datetime.datetime

property name
Returns a nicely formatted name based on available fields. Includes the volume number, chapter number, and chapter title if any one or more of them exist.

Returns Formatted name

Return type str

number: Optional[str]
The number of the chapter. None if the chapter is un-numbered (such as in an anthology).

Note: A chapter can have a number, a title, or both. If a chapter's number is None, it must have a title.

page_names: List[str]
A list of strings containing the filenames of the pages.

See also:

data_saver_page_names

async pages(*, *data_saver: bool = False*, *ssl_only: bool = False*) → List[str]
Get fully formatted page URLs.

Note: The given page URLs are only valid for a short timeframe. These URLs cannot be used for hotlinking.

Parameters

- **data_saver** (*bool*) – Whether or not to return the pages for the data saver URLs. Defaults to False.
- **ssl_only** (*bool*) – Whether or not the given URL has port 443. Useful if your firewall blocks outbound connections to ports that are not port 443. Defaults to False.

  Note: This will lower the pool of available clients and can cause higher latencies.

Returns A list of valid URLs in the order of the pages.

Return type List[str]

parse(*data: Dict[str, Any]*)
Parse the data received from the server.

Parameters **data** (*Dict[str, Any]*) – The data from the server.

publish_time: datetime.datetime
A datetime.datetime representing the time the chapter was published.

See also:

created_at

Note: The datetime is **timezone aware** as it is parsed from an ISO-8601 string.

property sorting_number
Returns 0 if the chapter does not have a number, otherwise returns the chapter's number.

Returns A number usable for sorting.

Return type float

title: Optional[str]
The title of the chapter. None if the chapter does not have a title.

Note: A chapter can have a number, a title, or both. If a chapter's title is `None`, it must have a number.

transfer(*new_obj: _T*)
Transfer data from a new object to the current object.

Parameters **new_obj** (`Model`) – The new object. Should be the same type as the current model.

updated_at: Optional[datetime]
A `datetime.datetime` representing the last time the object was updated. May be None if the object was never updated after creation.

See also:

modified_at()

Note: The datetime is **timezone aware** as it is parsed from an ISO-8601 string.

user: asyncdex.models.user.User
The user that uploaded this chapter.

version: int
The version of the model.

volume: Optional[str]
The volume of the chapter. `None` if the chapter belongs to no volumes.

class asyncdex.models.**Group**(*client: MangadexClient*, ***, *id: Optional[str] = None*, *version: int = 0*, *data: Optional[Dict[str, Any]] = None*)
A *Model* representing an individual scanlation group.

New in version 0.3.

__eq__(*other: _T*) → bool
Check if two models are equal to each other.

Parameters **other** (`Model`) – Another model. Should be the same type as the model being compared.

Returns Whether or not the models are equal.

Return type bool

__ge__(*other: _T*) → bool
Compares the two object's creation times to find if the current model's creation time is greater than or equal to the other model's creation time.

New in version 0.3.

Parameters **other** (`DatetimeMixin`) – The other model.

Returns Whether or not the current model's creation time is greater than or equal to the other model's creation time.

Return type bool

__gt__(*other: _T*) → bool
Compares the two object's creation times to find if the current model's creation time is greater than the other model's creation time.

New in version 0.3.

Parameters **other** (DatetimeMixin) – The other model.

Returns Whether or not the current model's creation time is greater than the other model's creation time.

Return type bool

__hash__()
Return hash(self).

__le__(*other: _T*) → bool
Compares the two object's creation times to find if the current model's creation time is less than or equal to the other model's creation time.

New in version 0.3.

Parameters **other** (DatetimeMixin) – The other model.

Returns Whether or not the current model's creation time is less than or equal to the other model's creation time.

Return type bool

__lt__(*other: _T*) → bool
Compares the two object's creation times to find if the current model's creation time is less than the other model's creation time.

New in version 0.3.

Parameters **other** (DatetimeMixin) – The other model.

Returns Whether or not the current model's creation time is less than the other model's creation time.

Return type bool

__ne__(*other: _T*) → bool
Check if two models are not equal to each other.

Parameters **other** (Model) – Another model. Should be the same type as the model being compared.

Returns Whether or not the models are equal.

Return type bool

__str__() → str
Returns a string representation of the model, usually it's id.

chapters: **List[*Chapter*]**
A list of chapters uploaded by the group.

client: **MangadexClient**
The client that created this model.

created_at: **datetime**
A datetime.datetime representing the object's creation time.

See also:

modified_at()

Note: The datetime is **timezone aware** as it is parsed from an ISO-8601 string.

---

async fetch()
Fetch the data to complete any missing non-critical values.

Raises *InvalidID* if an object with the ID does not exist.

id: str
A UUID that represents this item.

leader: asyncdex.models.user.User
The user who created the group.

async load_chapters()
Shortcut method that calls `Client.batch_chapters()` with the chapters that belong to the group.

Roughly equivalent to:

```
await client.batch_chapters(*user.chapters)
```

members: List[asyncdex.models.user.User]
Users who are members of the group.

property modified_at
The last time the object was modified. This will return the creation time if the object was never updated after creation, or the modification time if it has.

See also:

created_at, *updated_at*

Returns

The last time the object was changed as a `datetime.datetime` object.

Note: The datetime is **timezone aware** as it is parsed from an ISO-8601 string.

Return type `datetime.datetime`

name: str
The name of the group.

parse(*data: Dict[str, Any]*)
Parse the data received from the server.

Parameters **data** (*Dict[str, Any]*) – The data from the server.

transfer(*new_obj: _T*)
Transfer data from a new object to the current object.

Parameters **new_obj** (`Model`) – The new object. Should be the same type as the current model.

updated_at: Optional[datetime]
A `datetime.datetime` representing the last time the object was updated. May be None if the object was never updated after creation.

See also:

modified_at()

Note: The datetime is **timezone aware** as it is parsed from an ISO-8601 string.

version: *int*
The version of the model.

class asyncdex.models.**Manga**(*client: MangadexClient*, ***, *id: Optional[str] = None*, *version: int = 0*, *data: Optional[Dict[str, Any]] = None*)
A *Model* representing an individual manga.

New in version 0.2.

__eq__(*other: _T*) → bool
Check if two models are equal to each other.

Parameters **other** (Model) – Another model. Should be the same type as the model being compared.

Returns Whether or not the models are equal.

Return type bool

__ge__(*other: _T*) → bool
Compares the two object's creation times to find if the current model's creation time is greater than or equal to the other model's creation time.

New in version 0.3.

Parameters **other** (DatetimeMixin) – The other model.

Returns Whether or not the current model's creation time is greater than or equal to the other model's creation time.

Return type bool

__gt__(*other: _T*) → bool
Compares the two object's creation times to find if the current model's creation time is greater than the other model's creation time.

New in version 0.3.

Parameters **other** (DatetimeMixin) – The other model.

Returns Whether or not the current model's creation time is greater than the other model's creation time.

Return type bool

__hash__()
Return hash(self).

__le__(*other: _T*) → bool
Compares the two object's creation times to find if the current model's creation time is less than or equal to the other model's creation time.

New in version 0.3.

Parameters **other** (DatetimeMixin) – The other model.

Returns Whether or not the current model's creation time is less than or equal to the other model's creation time.

Return type bool

__lt__(*other: _T*) → bool
Compares the two object's creation times to find if the current model's creation time is less than the other model's creation time.

New in version 0.3.

Parameters **other** (DatetimeMixin) – The other model.

Returns Whether or not the current model's creation time is less than the other model's creation time.

Return type bool

__ne__(*other: _T*) → bool
Check if two models are not equal to each other.

Parameters **other** (Model) – Another model. Should be the same type as the model being compared.

Returns Whether or not the models are equal.

Return type bool

__str__() → str
Returns a string representation of the model, usually it's id.

amazon_id: Optional[str]
The ID for the entry on Amazon, if it exists.

property amazon_url
Returns a formatted url for the manga's Amazon entry if it exists.

Note: While the MangaDex API currently returns fully formatted URLs for the *amazon_id* attribute, this may change in the future. This property will always return a fully formatted URL.

Returns A full URL or None if *amazon_id* is None.

Return type str

anilist_id: Optional[str]
The ID for the entry on Anilist, if it exists.

property anilist_url
Returns a formatted url for the manga's Anilist entry if it exists.

Returns A full URL or None if *anilist_id* is None.

Return type str

animeplanet_id: Optional[str]
The ID for the entry on AnimePlanet, if it exists.

property animeplanet_url
Returns a formatted url for the manga's AnimePlanet entry if it exists.

Returns A full URL or None if *animeplanet_id* is None.

Return type str

artists: List[asyncdex.models.author.Author]
A list of *Author* objects that represent the manga's artists.

See also:

authors

Note: In order to efficiently get all authors and artists in one go, use *load_authors()*.

authors: List[asyncdex.models.author.Author]
A list of *Author* objects that represent the manga's authors.

See also:

artists

Note: In order to efficiently get all authors and artists in one go, use *load_authors()*.

---

bookwalker_id: Optional[str]
The ID for the entry on Bookwalker, if it exists.

property bookwalker_url
Returns a formatted url for the manga's Bookwalker entry if it exists.

Returns A full URL or None if *bookwalker_id* is None.

Return type str

cdjapan_id: Optional[str]
The ID for the entry on CDJapan, if it exists.

property cdjapan_url
Returns a formatted url for the manga's CDJapan entry if it exists.

Note: While the MangaDex API currently returns fully formatted URLs for the *cdjapan_id* attribute, this may change in the future. This property will always return a fully formatted URL.

Returns A full URL or None if *cdjapan_id* is None.

Return type str

chapters: asyncdex.models.chapter.ChapterList
A *ChapterList* representing the chapters of the manga.

New in version 0.3.

client: MangadexClient
The client that created this model.

created_at: datetime
A `datetime.datetime` representing the object's creation time.

See also:

modified_at()

Note: The datetime is **timezone aware** as it is parsed from an ISO-8601 string.

---

demographic: *asyncdex.enum.Demographic*
The manga's demographic.

descriptions: *asyncdex.utils.DefaultAttrDict*[Optional[str]]
A *DefaultAttrDict* holding the descriptions of the manga.

Note: If a language is missing a description, `None` will be returned.

---

ebookjapan_id: Optional[str]
The ID for the entry on EbookJapan, if it exists.

property ebookjapan_url
Returns a formatted url for the manga's EbookJapan entry if it exists.

Note: While the MangaDex API currently returns fully formatted URLs for the *ebookjapan_id* attribute, this may change in the future. This property will always return a fully formatted URL.

Returns A full URL or None if *ebookjapan_id* is None.

Return type str

english_translation_url: Optional[str]
The URL for the official English translation of the manga, if it exists.

async fetch()
Fetch the data to complete any missing non-critical values.

Raises *InvalidID* if an object with the ID does not exist.

id: str
A UUID that represents this item.

kitsu_id: Optional[str]
The ID for the entry on Kitsu, if it exists.

property kitsu_url
Returns a formatted url for the manga's Kitsu entry if it exists.

Returns A full URL or None if *kitsu_id* is None.

Return type str

last_chapter: Optional[str]
The last chapter of the manga. `None` if it is not specified or does not exist.

Changed in version 0.3: Changed to a string in order to better match the API specification.

last_volume: Optional[str]
The last volume of the manga. `None` if it is not specified or does not exist.

Changed in version 0.3: Changed to a string in order to better match the API specification.

async load_authors()
Shortcut method that calls `Client.batch_authors()` with the authors and artists that belong to the manga.

Roughly equivalent to:

```
await client.batch_authors(*manga.authors, *manga.artists)
```

locked: bool
A locked manga. Usually means that chapter details cannot be modified.

mangaupdates_id: Optional[str]
The ID for the entry on MangaUpdates, if it exists.

property mangaupdates_url
Returns a formatted url for the manga's MangaUpdates entry if it exists.

Returns A full URL or None if *mangaupdates_id* is None.

Return type str

property modified_at
The last time the object was modified. This will return the creation time if the object was never updated after creation, or the modification time if it has.

See also:

created_at, *updated_at*

Returns

The last time the object was changed as a `datetime.datetime` object.

Note: The datetime is **timezone aware** as it is parsed from an ISO-8601 string.

Return type `datetime.datetime`

myanimelist_id: Optional[str]
The ID for the entry on MyAnimeList, if it exists.

property myanimelist_url
Returns a formatted url for the manga's MyAnimeList entry if it exists.

Returns A full URL or None if *myanimelist_id* is None.

Return type str

novelupdates_id: Optional[str]
The ID for the entry on NovelUpdates, if it exists.

property novelupdates_url
Returns a formatted url for the manga's NovelUpdates entry if it exists.

Returns A full URL or None if *novelupdates_id* is None.

Return type str

original_language: str
The original language that the manga was released in.

parse(*data: Dict[str, Any]*)
Parse the data received from the server.

Parameters **data** (*Dict[str, Any]*) – The data from the server.

rating: *asyncdex.enum.ContentRating*
The manga's content rating.

raw_url: Optional[str]
The URL for the official raws of the manga, if it exists.

status: *asyncdex.enum.MangaStatus*
The manga's status.

tags: List[asyncdex.models.tag.Tag]
A list of *Tag* objects that represent the manga's tags. A manga without tags will have an empty list.

titles: *asyncdex.utils.DefaultAttrDict*[*asyncdex.models.title.TitleList*]
A *DefaultAttrDict* holding the titles of the manga.

transfer(*new_obj: _T*)
Transfer data from a new object to the current object.

Parameters **new_obj** (Model) – The new object. Should be the same type as the current model.

updated_at: Optional[datetime]
A datetime.datetime representing the last time the object was updated. May be None if the object was never updated after creation.

See also:

modified_at()

Note: The datetime is **timezone aware** as it is parsed from an ISO-8601 string.

version: int
The version of the model.

year: Optional[int]
The year the manga started publication. May be None if publication hasn't started or is unknown.

class asyncdex.models.**Tag**(*client: MangadexClient*, ***, *id: Optional[str] = None*, *version: int = 0*, *data: Optional[Dict[str, Any]] = None*)
A *Model* representing a tag in a Manga.

New in version 0.2.

__eq__(*other: _T*) → bool
Check if two models are equal to each other.

Parameters **other** (Model) – Another model. Should be the same type as the model being compared.

Returns Whether or not the models are equal.

Return type bool

__hash__()
Return hash(self).

__ne__(*other: _T*) → bool
Check if two models are not equal to each other.

Parameters **other** (Model) – Another model. Should be the same type as the model being compared.

Returns Whether or not the models are equal.

Return type bool

__str__() → str
Returns a string representation of the model, usually it's id.

client: MangadexClient
The client that created this model.

descriptions: *asyncdex.utils.DefaultAttrDict*[Optional[str]]
A *DefaultAttrDict* holding the descriptions of the tag.

New in version 0.4.

Note: If a language is missing a description, None will be returned.

async fetch()
Fetch the data to complete any missing non-critical values.

Raises *InvalidID* if an object with the ID does not exist.

group: Optional[str]
The group that the tag belongs to.

New in version 0.4.

id: str
A UUID that represents this item.

names: asyncdex.utils.DefaultAttrDict[Optional[str]]
A *DefaultAttrDict* holding the names of the tag.

Note: If a language is missing a name, `None` will be returned.

parse(*data: Dict[str, Any]*)
Parse the data received from the server.

Parameters **data** (*Dict[str, Any]*) – The data from the server.

transfer(*new_obj: _T*)
Transfer data from a new object to the current object.

Parameters **new_obj** (Model) – The new object. Should be the same type as the current model.

version: int
The version of the model.

class asyncdex.models.**User**(*client: MangadexClient*, *, *id: Optional[str] = None*, *version: int = 0*, *data: Optional[Dict[str, Any]] = None*)
A *Model* representing an individual user.

New in version 0.3.

__eq__(*other: _T*) → bool
Check if two models are equal to each other.

Parameters **other** (Model) – Another model. Should be the same type as the model being compared.

Returns Whether or not the models are equal.

Return type bool

__hash__()
Return hash(self).

__ne__(*other: _T*) → bool
Check if two models are not equal to each other.

Parameters **other** (Model) – Another model. Should be the same type as the model being compared.

Returns Whether or not the models are equal.

Return type bool

__str__() → str
Returns a string representation of the model, usually it's id.

chapters: List[*Chapter*]
The chapters the user uploaded

client: MangadexClient
The client that created this model.

async fetch()
Fetch the data to complete any missing non-critical values.

Raises *InvalidID* if an object with the ID does not exist.

id: str
A UUID that represents this item.

async load_chapters()
Shortcut method that calls `Client.batch_chapters()` with the chapters that belong to the user.

Roughly equivalent to:

```
await client.batch_chapters(*user.chapters)
```

parse(*data: Dict[str, Any]*)
Parse the data received from the server.

Parameters **data** (*Dict[str, Any]*) – The data from the server.

transfer(*new_obj: _T*)
Transfer data from a new object to the current object.

Parameters **new_obj** (Model) – The new object. Should be the same type as the current model.

username: str
THe user's username.

version: int
The version of the model.

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 `MangadexClient.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.

exception asyncdex.exceptions.**Missing**(*attribute: str*, *model: Optional[str] = None*)
An exception raised if a response is missing a critical element for a model.

New in version 0.2.

Parameters **model** (*str*) – The name of the model that requires the attribute. Can be empty.

attribute: *str*
The name of the attribute that is missing.

exception asyncdex.exceptions.**InvalidID**(*id: str*, *model: Type[Model]*)
An exception raised if an invalid ID is given to any of the get_* methods representing that an item with this ID does not exist.

New in version 0.2.

id: *str*
The given ID

model: *Type[Model]*
The model that would have been returned had the ID been valid.

3.4 Enums

class asyncdex.enum.**Demographic**(*value*)
An Enum representing the various demographics. Source: https://api.mangadex.org/docs.html#section/Static-data/Manga-publication-demographic.

New in version 0.2.

JOSEI = 'josei'
A Josei Manga.

Changed in version 0.3: The typo for this field has been corrected.

NONE = 'none'
A manga without a demographic.

New in version 0.4.

SEINEN = 'seinen'
A Seinen Manga.

SHOUJO = 'shoujo'
A Shoujo Manga.

SHOUNEN = 'shounen'
A Shounen Manga.

> **Note:** In the developer documentation as of May 7, 2021, there is a typo in the word `Shounen`, where it is spelled without the `u`. However, the actual API will only recognize the variant including a `u`. For the library, both variations can be used for the enum.

class asyncdex.enum.**MangaStatus**(*value*)
An Enum representing the various statuses a manga can have. Source: https://api.mangadex.org/docs.html#section/Static-data/Manga-status

New in version 0.2.

> **Note:** The status of the manga does not dictate whether or not the chapter list will be stable. Scanlation teams may have not published all chapters up to the completion of updates, so the manga may still get new chapters, especially in different languages. The only way to know if a manga has actually finished updating is by checking if the "end chapter" is present in the current language. Even this is not a guarantee, as an author may add additional media accompanying the work, such as a extra page related to the manga on Twitter or Pixiv, especially for manga that are mainly published online. The labels shown for a manga's status should not dictate the policy for update checking, as they are only meant to be an aid for end users and not actually representative of the immutability of the manga's chapter list.

CANCELLED = 'cancelled'
A manga where the author has intentionally stopped publishing new chapters.

Changed in version 0.3: The MangaDex API changed the value from `abandoned` to `cancelled`. `MangaStatus.ABANDONED` will continue to represent the right value, but calling the enum with `abandoned` will not.

COMPLETED = 'completed'
A manga that has finished publication.

HIATUS = 'hiatus'
A manga where the author is on a known hiatus.

ONGOING = 'ongoing'
A manga that is actively being published, in volume format, in a magazine like Weekly Shonen, or online.

class asyncdex.enum.**FollowStatus**(*value*)
An Enum representing the status that the user has marked the manga has. Source: https://api.mangadex.org/docs.html#section/Static-data/Manga-reading-status

New in version 0.2.

COMPLETED = 'completed'
A manga that the user has marked as completed.

> **Warning:** When a manga is marked as completed, the MangaDex API drops all chapter read markers. Setting a manga as completed **will** result in the deletion of data. Be very careful!

DROPPED = 'dropped'
A manga that the user has marked as dropped.

ON_HOLD = 'on_hold'
A manga that the user has marked as "on hold".

PLAN_TO_READ = 'plan_to_read'
A manga that the user has marked as "plan to read".

READING = 'reading'
A manga that the user has marked as reading.

RE_READING = 're_reading'
A manga that the user has marked as rereading.

class asyncdex.enum.**ContentRating**(*value*)
An Enum representing the content in a manga. Source: https://api.mangadex.org/docs.html#section/Static-data/Manga-content-rating

New in version 0.2.

EROTICA = 'erotica'
A manga that is erotica.

Note: This type of content represents content tagged with the `Smut` tag.

PORNOGRAPHIC = 'pornographic'
A manga that is pornographic.

Note: This type of content was the only type of content that MangaDex's old 18+ filter used to block. This type of content was also the type of content that old MangaDex APIs used to call "hentai".

SAFE = 'safe'
A manga that is safe.

Note: This is the only content rating that means a manga is safe for work. All other values are not safe for work (NSFW).

SUGGESTIVE = 'suggestive'
A manga that is suggestive.

Note: This type of content represents content tagged with the `Ecchi` tag.

---

class asyncdex.enum.**Visibility**(*value*)
An enum representing the visibility of an `CustomList`. Source: https://api.mangadex.org/docs.html#section/Static-data/CustomList-visibility

New in version 0.2.

PRIVATE = 'private'
A private `CustomList`.

PUBLIC = 'public'
A public `CustomList`.

class asyncdex.enum.**Relationship**(*value*)
An enum representing the different types of relationship types. Source: https://api.mangadex.org/docs.html#section/Static-data/Relationship-types

New in version 0.2.

ARTIST = 'artist'
A *Author* resource.

AUTHOR = 'author'
A *Author* resource.

CHAPTER = 'chapter'
A *Chapter* resource.

CUSTOM_LIST = 'custom_list'
A `CustomList` resource.

MANGA = 'manga'
A *Manga* resource.

SCANLATION_GROUP = 'scanlation_group'
A *Group* resource.

TAG = 'tag'
A *Tag* resource.

USER = 'user'
A *User* resource.

class asyncdex.enum.**DuplicateResolutionAlgorithm**(*value*)
An enum representing the various methods of resolving duplicate chapters in the same language.

New in version 0.3.

Note: The filtering algorithms are short-circuiting, meaning that once the chapters for a certain chapter number is lowered down to one item, it will be returned.

Operation order:

1. Previous group
2. Specific Group
3. Specific User
4. Creation Date ascending/descending/Views ascending/descending

---

Note: It is an error to specify more than one of the lowest-priority operations, since they all return only one value. Doing so will raise an error.

CREATION_DATE_ASC = 4
A resolution strategy that will select the chapter that was created first.

See also:

CREATION_DATE_DESC

CREATION_DATE_DESC = 5
A resolution strategy that will select the chapter that was created last.

See also:

CREATION_DATE_ASC

PREVIOUS_GROUP = 1
A resolution strategy that attempts to use the same group for the chapter as the previous chapter. This needs an accompanying strategy to determine the initial group.

See also:

SPECIFIC_GROUP

SPECIFIC_GROUP = 2
A resolution strategy that attempts to only select certain groups. This needs an accompanying strategy for chapters where the group is not present.

See also:

SPECIFIC_USER

SPECIFIC_USER = 3
A resolution strategy that attempts to only select chapters by certain users. This needs an accompanying strategy for chapters where the user ia not present.

See also:

SPECIFIC_GROUP

VIEWS_ASC = 6
A resolution strategy that will select the chapter with the least views.

> **Warning:** This is not implemented yet as the API does not return view counts.

See also:

VIEWS_DESC

VIEWS_DESC = 7
A resolution strategy that will select the chapter with the most views.

> **Warning:** This is not implemented yet as the API does not return view counts.

See also:

VIEWS_ASC

class asyncdex.enum.**OrderDirection**(*value*)
An enum representing the various directions that can be used for ordering a list of items.

New in version 0.4.

ASCENDING = 'asc'
Order items from smallest to largest.

DESCENDING = 'desc'
Order items from largest to smallest.

class asyncdex.enum.**TagMode**(*value*)
An enum representing the various ways tag inclusion/exclusion can be read by the server.

New in version 0.4.

AND = 'AND'
Manga is included/excluded only if **all** tags are present.

OR = 'OR'
Manga is included/excluded if **any** tag is present.

3.5 Constants

asyncdex.constants.**invalid_folder_name_regex**
The regex for invalid folder names.

Contains:

- **Windows/MacOS/Linux restricted characters:**
  - `<`
  - `>`
  - `:`
  - `"`
  - `/`
  - `\`
  - `|`
  - `?`
  - `*`
- All control characters from `0x0` through `0x31` inclusive.
- **Windows restricted filename:**
  - `CON`
  - `PRN`
  - `AUX`
  - `NUL`
  - `COM1`
  - `COM2`
  - `COM3`
  - `COM4`
  - `COM5`
  - `COM6`
  - `COM7`
  - `COM8`
  - `COM9`
  - `LPT1`
  - `LPT2`
  - `LPT3`
  - `LPT4`

- LPT5
- LPT6
- LPT7
- LPT8
- LPT9

Source: https://stackoverflow.com/a/31976060/12248328

New in version 0.3.

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.

Changed in version 0.3.

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.

Changed in version 0.4: `mdah` renamed to `md@h`.

3.6 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.

__repr__() → str
Provide a string representation of the object.

Returns The string representation

Return type str

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 *PathRatelimit*

3.7 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

class asyncdex.utils.**AttrDict**

A dict where keys can be accessed by attributes.

New in version 0.2.

__getattr__(*item: str*) → _VT

Get a key of the dictionary by calling the attribute representing it.

Parameters **item** (*str*) – The key to get.

Returns The value that is held inside the dictionary.

Return type Any

Raises KeyError if the attribute does not exist in the key.

__repr__() → str

Provide a string representation of the object.

Returns The string representation

Return type str

__setattr__(*key: str*, *value: _VT*)

Sets a key of the dictionary.

Parameters

- **key** (*str*) – The key to set.
- **value** (*Any*) – The value for the key.

first() → _VT

Return the first entry in the dictionary.

Returns The first entry.

Raises KeyError if there are no entries in the dictionary.

Return type Any

class asyncdex.utils.**DefaultAttrDict**(*mapping_or_iterable: Optional[Union[Mapping[str, _VT], Iterable[Tuple[str, _VT]]]] = None*, ***, *default: Callable[], _VT]*)

An *AttrDict* with a default.

New in version 0.2.

__missing__(*key: str*) → _VT

Apply the default if a key does not exist.

Parameters **key** (*str*) – The key that is missing

Returns The new default

Return type Any

default

A callable that accepts no arguments and returns an instance of the value's class.

asyncdex.utils.**copy_key_to_attribute**(*source_dict: dict*, *key: str*, *obj: Any*, *attribute_name: Optional[str] = None*, ***, *default: Any = Sentinel*, *transformation: Optional[Callable[[str], Any]] = None*)

Copies the value of a dictionary's key to an object.

New in version 0.2.

Parameters

- **source_dict** (*dict*) – The dictionary with the key and value.
- **key** (*str*) – The key that has the value.
- **obj** (*Any*) – The object to set the attribute of.
- **attribute_name** (*str*) – The name of the attribute to set if the name of the key and the name of the attribute are different.
- **default** (*Any*) – A default value to add if the value is not found.
- **transformation** (*Callable[[str], Any]*) – A callable that will be executed on the value of the key. It should accept a `str` and can return anything.

asyncdex.utils.**parse_relationships**(*data: dict*, *obj: Model*)

Parse the relationships available in a model.

New in version 0.2.

Changed in version 0.3: Added support for *Chapter*, `User, and :class:`.Group` objects.

Parameters

- **data** (*dict*) – The raw data received from the API.
- **obj** (`Model`) – The object to add the models to.

class asyncdex.models.mixins.**DatetimeMixin**

A mixin for models with `created_at` and `updated_at` attributes.

New in version 0.2.

__ge__(*other: _T*) → bool

Compares the two object's creation times to find if the current model's creation time is greater than or equal to the other model's creation time.

New in version 0.3.

Parameters **other** (`DatetimeMixin`) – The other model.

Returns Whether or not the current model's creation time is greater than or equal to the other model's creation time.

Return type bool

__gt__(*other: _T*) → bool
Compares the two object's creation times to find if the current model's creation time is greater than the other model's creation time.

New in version 0.3.

Parameters **other** (DatetimeMixin) – The other model.

Returns Whether or not the current model's creation time is greater than the other model's creation time.

Return type bool

__le__(*other: _T*) → bool
Compares the two object's creation times to find if the current model's creation time is less than or equal to the other model's creation time.

New in version 0.3.

Parameters **other** (DatetimeMixin) – The other model.

Returns Whether or not the current model's creation time is less than or equal to the other model's creation time.

Return type bool

__lt__(*other: _T*) → bool
Compares the two object's creation times to find if the current model's creation time is less than the other model's creation time.

New in version 0.3.

Parameters **other** (DatetimeMixin) – The other model.

Returns Whether or not the current model's creation time is less than the other model's creation time.

Return type bool

created_at: **datetime.datetime**
A datetime.datetime representing the object's creation time.

See also:

modified_at()

Note: The datetime is **timezone aware** as it is parsed from an ISO-8601 string.

---

property modified_at
The last time the object was modified. This will return the creation time if the object was never updated after creation, or the modification time if it has.

See also:

created_at, *updated_at*

Returns

The last time the object was changed as a datetime.datetime object.

Note: The datetime is **timezone aware** as it is parsed from an ISO-8601 string.

Return type datetime.datetime

updated_at: Optional[datetime.datetime]
A datetime.datetime representing the last time the object was updated. May be None if the object was never updated after creation.

See also:

modified_at()

Note: The datetime is **timezone aware** as it is parsed from an ISO-8601 string.

class asyncdex.models.title.**TitleList**(*iterable=()*, */*)
An object representing a list of titles.

New in version 0.2.

__repr__() → str
Provide a string representation of the object.

Returns The string representation

Return type str

property primary
Returns the primary title for the language if it exists or else returns None.

Returns The first title in the list.

Return type str

class asyncdex.models.**ChapterList**(*manga: Manga*, ***, *entries: Optional[Iterable[asyncdex.models.chapter.Chapter]] = None*)
An object representing a list of chapters from a manga.

New in version 0.3.

Parameters **entries** (*Iterable[Chapter]*) – Pre-fill the ChapterList with the given entries.

__repr__() → str
Provide a string representation of the object.

Returns The string representation

Return type str

async download_all(***, *skip_bad: bool = True*, *folder_format: str = '{manga}/{chapter_num}{separator}{title}'*, *file_format: str = '{num}'*, *as_bytes_list: bool = False*, *overwrite: bool = True*, *retries: int = 3*, *use_data_saver: bool = False*, *ssl_only: bool = False*) → Dict[asyncdex.models.chapter.Chapter, Optional[List[str]]]
Download all chapters in the list.

New in version 0.4.

Parameters

- **skip_bad** (*bool*) – Whether or not to skip bad chapters. Defaults to True.

- **folder_format** (*str*) – The format of the folder to create for the chapter. The folder can already be existing. The default format is `{manga}/{chapter_num}{separator}{chapter_title}`.

  Note: Specify `.` if you want to save the pages in the current folder.

  Available variables:

  - `{manga}`: The name of the manga. If the chapter's manga object does not contain a title object, it will be fetched.
  - `{chapter_num}`: The number of the chapter, if it exists.
  - `{separator}`: A separator if both the chapter's number and title exists.
  - `{title}`: The title of the chapter, if it exists.

- **file_format** (*str*) – The format of the individual image file names. The default format is `{num}`.

  Note: The file extension is applied automatically from the real file name. There is no need to include it.

  Available variables:

  - `{num}`: The numbering of the image files starting from 1. This respects the order the images are in inside of *page_names*.
  - `{num0}`: The same as `{num}` but starting from 0.
  - `{name}`: The actual filename of the image from *page_names*, without the file extension.

- **as_bytes_list** (*bool*) – Whether or not to return the pages as a list of raw bytes. Setting this parameter to `True` will ignore the value of the `folder_format` parameter.
- **overwrite** (*bool*) – Whether or not to override existing files with the same name as the page. Defaults to `True`.
- **retries** (*int*) – How many times to retry a chapter if a MD@H node does not let us download the pages. Defaults to `3`.
- **use_data_saver** (*bool*) – Whether or not to use the data saver pages or the normal pages. Defaults to `False`.
- **ssl_only** (*bool*) – Whether or not the given URL has port `443`. Useful if your firewall blocks outbound connections to ports that are not port `443`. Defaults to `False`.

  Note: This will lower the pool of available clients and can cause higher download times.

Raises `aiohttp.ClientResponseError` if there is an error after all retries are exhausted.

Returns A dictionary mapping consisting of *Chapter* objects as keys and the data from that chapter's *download_chapter()* method. If `skip_bad` is True, chapters with exceptions will have `None` instead of a list of bytes.

Return type List[Optional[List[bytes]]]

filter(*, *locales: Optional[List[str]] = None*, *creation_time: Optional[asyncdex.utils.Interval[datetime.datetime]] = None*, *update_time: Optional[asyncdex.utils.Interval[datetime.datetime]] = None*, *publish_time: Optional[asyncdex.utils.Interval[datetime.datetime]] = None*, *views: Optional[asyncdex.utils.Interval[int]] = None*, *has_number: Optional[bool] = None*, *chapter_number_range: Optional[asyncdex.utils.Interval[float]] = None*, *chapter_numbers: Optional[asyncdex.utils.InclusionExclusionPair[Optional[float]]] = None*, *remove_duplicates: bool = False*, *duplicate_strategy: Optional[List[asyncdex.enum.DuplicateResolutionAlgorithm]] = None*, *duplicate_strategy_groups: Optional[List[asyncdex.models.group.Group]] = None*, *duplicate_strategy_users: Optional[List[asyncdex.models.user.User]] = None*, *groups: Optional[asyncdex.utils.InclusionExclusionPair[asyncdex.models.group.Group]] = None*, *users: Optional[asyncdex.utils.InclusionExclusionPair[asyncdex.models.user.User]] = None*) → asyncdex.models.chapter.ChapterList

Filter the chapter list and return a new *ChapterList*. Calling this method without specifying any additional filtering mechanisms will return a shallow copy of the list.

The order of the filter will be as follows:

1. Filter the datetimes first
2. Filter by the intervals
3. Filter by the inclusion and exclusion pairs
4. Filter duplicates

Parameters

- **locales** (*List[str]*) – The locales that should be present in the chapters.
- **creation_time** (*Interval[datetime]*) – An *Interval* representing the bounds of the chapter's creation time. *Interval.min* will select all chapters created **after** the given time, and *Interval.max* will select all chapters created **before** the given time.

Note: The datetime objects needs to be a non-timezone aware datetime in UTC time. A datetime in any timezone can be converted to a naive UTC timezone by:

```
from datetime import timezone
# dt is the datetime object.
utc_naive = dt.astimezone(timezone.utc).replace(tzinfo=None)
```

Example intervals:

```
from asyncdex import Interval
min_interval = Interval(min=datetime.datetime(2021, 1, 1))
max_interval = Interval(max=datetime.datetime(2021, 1, 1))
both = Interval(datetime.datetime(2021, 1, 1), datetime.
↪datetime(2021, 5, 1))
```

- **update_time** (*Interval[datetime]*) – An *Interval* representing the bounds of the chapter's update time. *Interval.min* will select all chapters updated **after** the given time, and *Interval.max* will select all chapters updated **before** the given time.

Note: The datetime objects needs to be a non-timezone aware datetime in UTC time. A datetime in any timezone can be converted to a naive UTC timezone by:

```
from datetime import timezone
# dt is the datetime object.
utc_naive = dt.astimezone(timezone.utc).replace(tzinfo=None)
```

Example intervals:

```
from asyncdex import Interval
min_interval = Interval(min=datetime.datetime(2021, 1, 1))
max_interval = Interval(max=datetime.datetime(2021, 1, 1))
both = Interval(datetime.datetime(2021, 1, 1), datetime.
↪datetime(2021, 5, 1))
```

- **publish_time** (*Interval[datetime]*) – An *Interval* representing the bounds of the chapter's publish time. *Interval.min* will select all chapters published **after** the given time, and *Interval.max* will select all chapters published **before** the given time.

Note: The datetime objects needs to be a non-timezone aware datetime in UTC time. A datetime in any timezone can be converted to a naive UTC timezone by:

```
from datetime import timezone
# dt is the datetime object.
utc_naive = dt.astimezone(timezone.utc).replace(tzinfo=None)
```

Example intervals:

```
min_interval = Interval(min=datetime.datetime(2021, 1, 1))
max_interval = Interval(max=datetime.datetime(2021, 1, 1))
both = Interval(datetime.datetime(2021, 1, 1), datetime.
↪datetime(2021, 5, 1))
```

- **views** (*Interval[int]*) – An *Interval* of the views that a manga can have.

Warning: The mangadex API does not return views yet, so specifying something for this parameter will result in `NotImplementedError` being raised.

Example intervals:

```
from asyncdex import Interval
min_interval = Interval(min=100)
max_interval = Interval(max=25000)
both = Interval(100, 25000)
```

- **has_number** (*bool*) – Only select chapters with valid numbers.
- **chapter_number_range** (*Interval[float]*) – An *Interval* of the number of the chapter.

Note: Chapters without a number will be given a provisional number of 0 when sorted.

Example intervals:

```
from asyncdex import Interval
min_interval = Interval(min=2)
max_interval = Interval(max=20.5)
both = Interval(2, 20.5)
```

- **chapter_numbers** (*InclusionExclusionPair[float]*) – An *InclusionExclusionPair* denoting the chapter numbers that are either included or excluded.

  Note: Chapters without a number will be given a provisional number of 0 when sorted.

  Example inclusion/exclusion pairs:

```
from asyncdex import InclusionExclusionPair
include = InclusionExclusionPair(include=[5, 6])
excluse = InclusionExclusionPair(exclude=[7, 8, 9.5])
```

- **remove_duplicates** (*bool*) – Whether or not to remove duplicate chapters, ie chapters with the same chapter number.

  Note: This will not take locales into consideration. Make sure to specify a locale in the `locales` parameter if you want duplicates filtered for a specific locale.

- **duplicate_strategy** (*List[DuplicateResolutionAlgorithm]*) – The list of strategies used to resolve duplicates. See the values in *DuplicateResolutionAlgorithm* to find the possible algorithms. By default, the strategy of choosing the previous group and the strategy of choosing the first chapter chronologically when there is no previous group will be used.

  Note: If an adequate strategy is not found for dealing with certain chapters, the fallback mechanism of selecting the chapter that was created first will be used.

- **duplicate_strategy_groups** (*List[Group]*) – The groups to use for *DuplicateResolutionAlgorithm.SPECIFIC_GROUP*.

  Note: If the group is not present in all the chapters for a specific number, an alternate resolution algorithm will be used. Use the `include_groups` param if you only want chapters from that group.

- **duplicate_strategy_users** (*List[User]*) – The users to use for *DuplicateResolutionAlgorithm.SPECIFIC_USER*.

  Note: If the user is not present in all the chapters for a specific number, an alternate resolution algorithm will be used. Use the `include_users` param if you only want chapters from that user.

- **users** (*InclusionExclusionPair[User]*) – An *InclusionExclusionPair* denoting the users to include/exclude from the listing.

- **groups** (*InclusionExclusionPair[Group]*) – An *InclusionExclusionPair* denoting the groups to include/exclude from the listing.

Returns

A filtered *ChapterList*.

Note: The filtered list is not cached in *Manga.chapters*.

Return type *ChapterList*

async get(*, *locales: Optional[List[str]] = None*, *created_after: Optional[datetime.datetime] = None*, *updated_after: Optional[datetime.datetime] = None*, *published_after: Optional[datetime.datetime] = None*)

Gets the list of chapters.

Parameters

- **locales** (*List[str]*) – The locales to filter by.
- **created_after** (*datetime*) – Get chapters created after this date.

Note: The datetime object needs to be in UTC time. It does not matter if the datetime if naive or timezone aware.

- **updated_after** (*datetime*) – Get chapters updated after this date.

Note: The datetime object needs to be in UTC time. It does not matter if the datetime if naive or timezone aware.

- **published_after** (*datetime*) – Get chapters published after this date.

Note: The datetime object needs to be in UTC time. It does not matter if the datetime if naive or timezone aware.

---

manga: *Manga*

The *Manga* that this chapter list belongs to.

sort(*, *key: Optional[Callable[[asyncdex.models.chapter.Chapter], Any]] = None*, *reverse: bool = False*)

Sort the ChapterList. This uses a natural sorting algorithm to sort the chapters.

Parameters

- **key** (*Callable[[Chapter], Any]*) – An optional key if you want to override the sorting key used by the class.
- **reverse** (*bool*) – Whether or not to reverse the list.

class asyncdex.models.pager.**Pager**(*url: str*, *model: Type[_ModelT]*, *client: MangadexClient*, ***, *params: Optional[MutableMapping[str, Any]] = None*, *limit_size: int = 100*, *limit: Optional[int] = None*)

A pager object which automatically paginates responses with an offset and limit combo.

New in version 0.3.

Parameters **limit_size** (*int*) – The maximum limit for each request. Defaults to 100.

__aiter__() → AsyncIterator[_ModelT]
Return an async iterator (itself)

Returns The Pager class.

Return type *Pager*

async __anext__() → _ModelT
Return a model from the queue. If there are no items remaining, a request is made to fetch the next set of items.

Changed in version 0.4: This method will no longer hang to complete all requests.

Returns The new model.

Return type *Model*

__repr__() → str
Provide a string representation of the object.

Returns The string representation

Return type str

async as_list() → List[_ModelT]
Returns all items in the Pager as a list.

client: MangadexClient
The client that is associated with the Pager.

limit: Optional[int]
The Pager will only return up to these many items.

New in version 0.4.

model: Type[_ModelT]
A subclass of *Model* to transform the results into.

params: MutableMapping[str, Any]
Additional params to include in every request.

url: str
The URL to paginate against.

class asyncdex.utils.**Interval**(*min: Optional[_T] = None*, *max: Optional[_T] = None*, *inclusive: bool = True*)
A class representing an interval.

New in version 0.3.

inclusive: bool = True
Whether or not the interval includes the min and max values or only values after and before respectively are considered.

max: Optional[_T] = None
The maximum value of the interval.

min: Optional[_T] = None
The minimum value of the interval.

class asyncdex.utils.**InclusionExclusionPair**(*include: List[_T] = <factory>*, *exclude: List[_T] = <factory>*)
A class representing an inclusion and exclusion pair.

New in version 0.3.

Note: It is an error to both include and exclude something.

---

exclude: List[_T]
Values that should not be present.

include: List[_T]
Values that should be present.

matches_include_exclude_pair(*item: _T*) → bool
Returns whether or not the item is inside the include and exclude pairs.

Parameters **item** (*Any*) – The item to check.

Returns Whether or not it matches the given bounds (in the include list or not in the exclude list).

Return type bool

asyncdex.utils.**return_date_string**(*datetime_obj: datetime.datetime*)
Get a representation of a datetime object suitable for the MangaDex API.

New in version 0.3.

Changed in version 0.4: The method was changed from a private method to a seperate utility.

Parameters **datetime_obj** (*datetime*) – The datetime object.

Returns A string representation suitable for the API.

Return type str

class asyncdex.models.tag.**TagDict**
An object representing a dictionary of tag UUIDs to tag objects.

New in version 0.4.

groups() → Dict[str, List[asyncdex.models.tag.Tag]]
Categorizes the tags contained into a dictionary of the groups the tags belong to.

Returns A dictionary of group name to the list of tags that contain the name.

Return type Dict[str, List[*Tag*]]

class asyncdex.list_orders.**AuthorListOrder**(*name: Optional[asyncdex.enum.OrderDirection] = None*)
An object representing the various options for ordering a author list returned from `Client.get_authors()`.

New in version 0.4.

name: Optional[*asyncdex.enum.OrderDirection*] = None
The name of an author.

class asyncdex.list_orders.**ChapterListOrder**(*creation_time: Optional[asyncdex.enum.OrderDirection] = None*, *update_time: Optional[asyncdex.enum.OrderDirection] = None*, *publish_time: Optional[asyncdex.enum.OrderDirection] = None*, *title: Optional[asyncdex.enum.OrderDirection] = None*, *volume: Optional[asyncdex.enum.OrderDirection] = None*, *number: Optional[asyncdex.enum.OrderDirection] = None*)

An object representing the various options for ordering a chapter list returned from `Client.get_chapters()`.

New in version 0.4.

creation_time: Optional[*asyncdex.enum.OrderDirection*] = None
The time a chapter was created.

number: Optional[*asyncdex.enum.OrderDirection*] = None
The chapter's number.

publish_time: Optional[*asyncdex.enum.OrderDirection*] = None
The time a chapter was published.

title: Optional[*asyncdex.enum.OrderDirection*] = None
The title of the chapter[?].

update_time: Optional[*asyncdex.enum.OrderDirection*] = None
The time a chapter was updated.

volume: Optional[*asyncdex.enum.OrderDirection*] = None
The chapter's volume.

class asyncdex.list_orders.**GroupListOrder**(*name: Union[asyncdex.enum.OrderDirection, NoneType] = None*)

name: Optional[*asyncdex.enum.OrderDirection*] = None
The name of the scanlation group[?].

class asyncdex.list_orders.**MangaListOrder**(*creation_time: Optional[asyncdex.enum.OrderDirection] = None*, *update_time: Optional[asyncdex.enum.OrderDirection] = None*, *titles: Optional[asyncdex.enum.OrderDirection] = None*, *year: Optional[asyncdex.enum.OrderDirection] = None*)

An object representing the various options for ordering a manga list returned from `Client.search()`.

New in version 0.4.

creation_time: Optional[*asyncdex.enum.OrderDirection*] = None
The time a manga was created.

titles: Optional[*asyncdex.enum.OrderDirection*] = None
The titles of a manga[?].

update_time: Optional[*asyncdex.enum.OrderDirection*] = None
The time a manga was updated.

year: Optional[*asyncdex.enum.OrderDirection*] = None
The year a manga was published.

See also:

Manga.year

3.8 References

-

INDEX

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

Y