Mastodon.py Documentation

Release 2.1.1

Lorenz Diener

Aug 19, 2025

INTRODUCTION

CHAPTER

ONE

USAGE

Register your app! This only needs to be done once (per server, or when distributing rather than hosting an application, most likely per device and server). Uncomment the code and substitute in your information:

```
from mastodon import Mastodon

'''
Mastodon.create_app(
    'pytooterapp',
    api_base_url = 'https://mastodon.social',
    to_file = 'pytooter_clientcred.secret'
)
'''
```

Then, log in. This can be done every time your application starts, or you can use the persisted information:

```
from mastodon import Mastodon

mastodon = Mastodon(client_id = 'pytooter_clientcred.secret',)
print(mastodon.auth_request_url())

# open the URL in the browser and paste the code you get
mastodon.log_in(
    code=input("Enter the OAuth authorization code: "),
    to_file="pytooter_usercred.secret"
)
```

Note that previous versions of Mastodon allowed logging in with username and password - unfortunately, due to security concerns, Mastodon has started requiring OAuth starting with version 4.4.0. If you're building a bot, you may with to instead just generate a token in the UI (On Mastodon: your-server.com/settings/applications) and use it directly

To post, create an actual API instance:

```
from mastodon import Mastodon

mastodon = Mastodon(access_token = 'pytooter_usercred.secret')
mastodon.toot('Tooting from Python using #mastodonpy !')
```

CHAPTER

TWO

INTRODUCTION

Mastodon is an ActivityPub-based Twitter-like federated social network node. It has an API that allows you to interact with its every aspect. This is a simple Python wrapper for that API, provided as a single Python module.

Mastodon.py aims to implement the complete public Mastodon API. As of this time, it is feature complete for Mastodon version 4.4.3. The Mastodon compatible API layers of various other pieces of software as well as forks, while not an official target, should also be basically compatible, and Mastodon.py does make some allowances for behaviour that isn't strictly like that of Mastodon, and attempts to support extensions to the API.

Some usage examples (not neccesarily following app development best practices, but enough to get you started if you learn best by example) can be found at https://github.com/halcy/MastodonpyExamples

CHAPTER

THREE

ACKNOWLEDGEMENTS

Mastodon.py contains work by a large number of contributors, many of which have put significant work into making it a better library. You can find some information about who helped with which particular feature or fix in the changelog.

CHAPTER

FOUR

RESEARCH USE AND CITING

If you use Mastodon.py in your research, please cite it according to the latest CITATION.cff from the repository:

https://github.com/halcy/Mastodon.py/blob/master/CITATION.cff

As a personal request, It is important to me to ask you to make sure that the subjects of your research - fediverse users - are alright with the research you are doing on them and/or that you have secured the approval of your institutions ethics board.

4.1 General information

4.1.1 Rate limiting

Mastodon's API rate limits per user account. By default, the limit is 300 requests per 5 minute time slot. This can differ from instance to instance and is subject to change. Mastodon.py has three modes for dealing with rate limiting that you can pass to the constructor, "throw", "wait" and "pace", "wait" being the default.

In "throw" mode, Mastodon.py makes no attempt to stick to rate limits. When a request hits the rate limit, it simply throws a *MastodonRateLimitError*. This is for applications that need to handle all rate limiting themselves (i.e. interactive apps), or applications wanting to use Mastodon.py in a multi-threaded context ("wait" and "pace" modes are not thread safe).

Note

Rate limit information is available on the *Mastodon* object for applications that implement their own rate limit handling.

`Mastodon.ratelimit_remaining`
: Number of requests allowed until the next reset.

`Mastodon.ratelimit_reset`
: Time at which the rate limit will next be reset, as a POSIX timestamp.

`Mastodon.ratelimit_limit`
: Total number of requests allowed between resets. Typically 300.

`Mastodon.ratelimit_lastcall`
: Time at which these values have last been seen and updated, as a POSIX timestamp.

In "wait" mode, once a request hits the rate limit, Mastodon.py will wait until the rate limit resets and then try again, until the request succeeds or an error is encountered. This mode is for applications that would rather just not worry about rate limits much, don't poll the API all that often, and are okay with a call sometimes just taking a while.

In "pace" mode, Mastodon.py will delay each new request after the first one such that, if requests were to continue at the same rate, only a certain fraction (set in the constructor as *ratelimit_pacefactor*) of the rate limit will be used up. The fraction can be (and by default, is) greater than one. If the rate limit is hit, "pace" behaves like "wait". This mode is probably the most advanced one and allows you to just poll in a loop without ever sleeping at all yourself. It is for applications that would rather just pretend there is no such thing as a rate limit and are fine with sometimes not being very interactive.

In addition to the per-user limit, there is a per-IP limit of 7500 requests per 5 minute time slot, and tighter limits on logins. Mastodon.py does not make any effort to respect these.

If your application requires many hits to endpoints that are available without logging in, do consider using Mastodon.py without authenticating to get the full per-IP limit.

4.1.2 Pagination

Many of Mastodon's API endpoints are paginated. What this means is that if you request data from them, you might not get all the data at once - instead, you might only get the first few results.

All endpoints that are paginated have four parameters: *since_id*, *max_id*, *min_id* and *limit*. *since_id* allows you to specify the smallest id you want in the returned data, but you will still always get the newest data, so if there are too many statuses between the newest one and *since_id*, some will not be returned. *min_id*, on the other hand, gives you statuses with that minimum id and newer, starting at the given id. *max_id*, similarly, allows you to specify the largest id you want. By specifying either min_id or *max_id* (generally, only one, not both, though specifying both is supported starting with Mastodon version 3.3.0) of them you can go through pages forwards and backwards.

On Mastodon mainline, you can, pass datetime objects as IDs when fetching posts, since the IDs used are Snowflake IDs and dates can be approximately converted to those. This is guaranteed to work on mainline Mastodon servers and very likely to work on all forks, but will **not** work on other servers implementing the API, like Pleroma, Misskey or Gotosocial. You should not use this if you want your application to be universally compatible. It's also relatively coarse-grained.

limit allows you to specify how many results you would like returned. Note that an instance may choose to return less results than you requested - by default, Mastodon will return no more than 40 statuses and no more than 80 accounts no matter how high you set the limit.

The responses returned by paginated endpoints contain a "link" header that specifies which parameters to use to get the next and previous pages. Mastodon.py parses these and stores them (if present) in the first (for the previous page) and last (for the next page) item of the returned list as _pagination_prev and _pagination_next. They are accessible only via attribute-style access. Note that this means that if you want to persist pagination info with your data, you'll have to take care of that manually (or persist objects, not just dicts).

There are convenience functions available for fetching the previous and next page of a paginated request as well as for fetching all pages starting from a first page. For details, see *fetch_next()*, *fetch_previous()*. and *fetch_remaining()*.

4.1.3 IDs and unpacking

Mastodon's API uses IDs in several places: User IDs, Toot IDs, ...

While debugging, it might be tempting to copy-paste IDs from the web interface into your code. This will not work, as the IDs on the web interface and in the URLs are not the same as the IDs used internally in the API, so don't do that.

ID unpacking

Wherever Mastodon.py expects an ID as a parameter, you can also pass a dict that contains an id - this means that, for example, instead of writing

```
mastodon.status_post("@somebody wow!", in_reply_to_id = toot["id"])
```

you can also just write

```
mastodon.status_post("@somebody wow!", in_reply_to_id = toot)
```

and everything will work as intended.

Snowflake IDs

Some IDs in Mastodon (such as those for statuses) are Snowflake IDs. These broadly correspond to times, with a low resolution, so it is possible to convert a time to a Snowflake ID and search for posts between two dates. Mastodon.py will do the conversion for you automatically when you pass a *datetime* object as the id.

Note that this functionality will *not* work on anything but Mastodon and forks, and that it is somewhat inexact due to the relatively low resolution.

4.1.4 Versioning

Mastodon.py will check if a certain endpoint is available before doing API calls. By default, it checks against the version of Mastodon retrieved on init(), or the version you specified. Mastodon.py can be set (in the constructor) to either check if an endpoint is available at all (this is the default) or to check if the endpoint is available and behaves as in the newest Mastodon version (with regards to parameters as well as return values). Version checking can also be disabled altogether. If a version check fails, Mastodon.py throws a *MastodonVersionError*.

Some functions need to check what version of Mastodon they are talking to. These will generally use a cached version to avoid sending a lot of pointless requests.

Many non-mainline forks have various different formats for their versions and they have different, incompatible ideas about how to report version. Mastodon.py tries its best to figure out what is going on, but success is not guaranteed.

With the following functions, you can make Mastodon.py re-check the server version or explicitly determine if a specific minimum Version is available. Long-running applications that aim to support multiple Mastodon versions should do this from time to time in case a server they are running against updated.

Mastodon.**retrieve_mastodon_version**() → str

Determine installed Mastodon version and set major, minor and patch (not including RC info) accordingly.

Returns the version string, possibly including rc info.

Mastodon.**verify_minimum_version**(*version_str: str*, *cached: bool = False*) → bool

Update version info from server and verify that at least the specified version is present.

If you specify "cached", the version info update part is skipped.

Returns True if version requirement is satisfied, False if not.

4.1.5 A brief note on block lists

Mastodon.py used to block three instances because these were particularly notorious for harassing trans people and I don't feel like I have an obligation to let software I distribute help people who want my friends to die. I don't want to be associated with that, at all.

Those instances are now all gone, any point that could have been has been made, and there is no list anymore.

> **Note**
>
> Trans rights are human rights.

4.2 Return values

Unless otherwise specified, all data is returned as Python dictionaries, matching the JSON format used by the API. Dates returned by the API are in ISO 8601 format and are parsed into Python datetime objects.

To make access easier, the dictionaries returned are wrapped by a class that adds read-only attributes for all dict values - this means that, for example, instead of writing

```
description = mastodon.account_verify_credentials()["source"]["note"]
```

you can also just write

```
description = mastodon.account_verify_credentials().source.note
```

and everything will work as intended. The class used for this is exposed as *AttribAccessDict*.

Since version 2.0.0, Mastodon.py is fully typed - there are now classes for all return types, and all functions have type hints. Note that the base class is still the AttribAccessDict - this means that you can still access all returned values as attributes, *even if a type does not define them*. Lists have been split into lists that can be paginated (i.e. that have pagination attributes) and those that cannot.

All return values can be converted from and to JSON using the *to_json()* and *from_json()* methods defined on the *mastodon.types_base.Entity* class.

4.3 Base types

class mastodon.types_base.**AttribAccessDict**(***kwargs*)

Base return object class for Mastodon.py.

Inherits from dict, but allows access via attributes as well as if it was a dataclass.

While the subclasses implement specific fields with proper typing, parsing and documentation, they all inherit from this class, and parsing is extremely permissive to allow for forward and backward compatibility as well as compatibility with other implementations of the Mastodon API.

This class can ALSO have pagination information attached, for paginating lists *inside* the object, because that's what Mastodon 4.3.0 does for groupee notifications. This is special cased in the class definition, though.

class mastodon.types_base.**PaginatableList**(**args*, ***kwargs*)

This is a list with pagination information attached.

It is returned by the API when a list of items is requested, and the response contains a Link header with pagination information.

class mastodon.types_base.**NonPaginatableList**(**args*, ***kwargs*)

This is just a list, without pagination information but annotated for serialization and deserialization.

class mastodon.types_base.**MaybeSnowflakeIdType**(*value*, **args*, ***kwargs*)

Represents, maybe, a snowflake ID.

Contains what a regular ID can contain (int or str) and will convert to int if containing an int or a str that naturally converts to an int (e.g. "123").

Can *also* contain a *datetime* which gets converted to a timestamp.

It's also just *maybe* a snowflake ID, because some implementations may not use those.

This may seem annoyingly complex, but the goal here is: 1) If constructed with some ID, return that ID unchanged, so equality and hashing work 2) Allow casting to int and str, just like a regular ID 3) Halfway transparently convert to and from datetime with the correct format for the server we're talking to

to_datetime() → datetime | None

Convert to datetime. This *can* fail because not every implementation of the masto API is guaranteed to actually use snowflake IDs where masto uses snowflake IDs, so it can in fact return None.

mastodon.types_base.**IdType**

IDs returned from Mastodon.py ar either primitive (int or str) or snowflake (still int or str, but potentially convertible to datetime), but also a datetime (which will get converted to a snowflake id).

class mastodon.types_base.**Entity**

Base class for everything returned by the API. This is a union of *AttribAccessDict* and *EntityList*.

Defines two methods: to_json(), and (static) from_json(), for serializing and deserializing to/from JSON.

to_json(*pretty=True*) → str

Serialize to JSON.

The returned JSON data includes type information and a version field.

static from_json(*json_str: str*) → *Entity*

Deserialize from JSON.

Parse a JSON string and cast to the to the appropriate type by using a special field that is added by serialization.

This *should* be safe to call on any JSON string (no less safe than json.loads), but I would still recommend to be very careful when using this on untrusted data and to check that the returned value matches your expectations.

There is currently a bug on specifically python 3.7 and 3.8 where the return value is not guaranteed to be of the right type. I will probably not fix this, since the versions are out of support, anyways. However, the data will still be loaded correctly.

mastodon.types_base.**EntityList**

Lists in Mastodon.py are either regular or paginatable, so this is a union of *NonPaginatableList* and *PaginatableList*.

4.4 Return types

class mastodon.return_types.**Account**(***kwargs*)

A user acccount, local or remote.

Example:

```
# Returns a Account object
mastodon.account(<account id>)
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Account/

id: *MaybeSnowflakeIdType*

The accounts id.

Version history:

- 0.1.0: added

username: **str**

The username, without the domain part.

Version history:

- 0.1.0: added

`acct: str`

The user's account name as username@domain (@domain omitted for local users).

Version history:

- 0.1.0: added

`display_name: str`

The user's display name.

Version history:

- 0.1.0: added

`discoverable: bool | None`

Indicates whether or not a user is visible on the discovery page. (nullable)

Version history:

- 3.1.0: added

`group: bool`

A boolean indicating whether the account represents a group rather than an individual.

Version history:

- 3.1.0: added

`locked: bool`

Denotes whether the account can be followed without a follow request.

Version history:

- 0.1.0: added

`created_at: datetime`

The accounts creation time.

Version history:

- 0.1.0: added
- 3.4.0: now resolves to midnight instead of an exact time

`following_count: int`

How many accounts this account follows.

Version history:

- 0.1.0: added

`followers_count: int`

How many followers this account has.

Version history:

- 0.1.0: added

`statuses_count: int`

How many statuses this account has created, excluding: 1) later deleted posts 2) direct messages / 'mentined users only' posts, except in earlier versions mastodon.

Version history:

- 0.1.0: added
- 2.4.2: no longer includes direct / mentioned-only visibility statuses

note: str

The users bio / profile text / 'note'.

Version history:

- 0.1.0: added

url: str

A URL pointing to this users profile page (can be remote). Should contain (as text): URL

Version history:

- 0.1.0: added

uri: str

Webfinger-resolvable URI for this account. Should contain (as text): URL

Version history:

- 4.2.0: added

avatar: str

URL for this users avatar, can be animated. Should contain (as text): URL

Version history:

- 0.1.0: added

header: str

URL for this users header image, can be animated. Should contain (as text): URL

Version history:

- 0.1.0: added

avatar_static: str

URL for this users avatar, never animated. Should contain (as text): URL

Version history:

- 1.1.2: added

header_static: str

URL for this users header image, never animated. Should contain (as text): URL

Version history:

- 1.1.2: added

moved: *Account* | None

If set, Account that this user has set up as their moved-to address. (optional)

Version history:

- 2.1.0: added

suspended: bool | None

Boolean indicating whether the user has been suspended. (optional)

Version history:

- 3.3.0: added

limited: `bool | None`

Boolean indicating whether the user has been silenced. (optional)

Version history:

- 3.5.3: added

bot: `bool`

Boolean indicating whether this account is automated.

Version history:

- 2.4.0: added

fields: *NonPaginatableList*[*AccountField*]

List of up to four (by default) AccountFields.

Version history:

- 2.4.0: added

emojis: *NonPaginatableList*[*CustomEmoji*]

List of custom emoji used in name, bio or fields.

Version history:

- 2.4.0: added

last_status_at: `datetime | None`

When the most recent status was posted. (nullable)

Version history:

- 3.0.0: added
- 3.1.0: now returns date only, no time

noindex: `bool | None`

Whether the local user has opted out of being indexed by search engines. (nullable)

Version history:

- 4.0.0: added

roles: *NonPaginatableList* `| None`

THIS FIELD IS DEPRECATED. IT IS RECOMMENDED THAT YOU DO NOT USE IT.

Deprecated. Was a list of strings with the users roles. Now just an empty list. Mastodon.py makes no attempt to fill it, and the field may be removed if Mastodon removes it. Use the *role* field instead. (optional, nullable)

Version history:

- 0.1.0: added
- 4.0.0: deprecated

role: *Role* `| None`

The users role. Only present for account returned from account_verify_credentials(). (optional)

Version history:

- 4.0.0: added

source: *CredentialAccountSource* | **None**

Additional information about the account, useful for profile editing. Only present for account returned from account_verify_credentials(). (optional)

Version history:

- 2.4.0: added

mute_expires_at: **datetime | None**

If the user is muted by the logged in user with a timed mute, when the mute expires. (nullable)

Version history:

- 3.3.0: added

indexable: **bool**

Boolean indicating whether public posts by this account should be searchable by anyone.

Version history:

- 4.2.0: added

hide_collections: **bool**

Boolean indicating whether a user has chosen to hide their network (followers/followed accounts).

Version history:

- 4.1.0: added

memorial: **bool | None**

Boolean indicating whether the account is an in-memoriam account. (optional)

Version history:

- 4.2.0: added

class mastodon.return_types.**AccountField**(***kwargs*)

A field, displayed on a users profile (e.g. “Pronouns”, “Favorite color”).

Example:

```
# Returns a AccountField object
mastodon.account(<account id>).fields[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Account/

name: **str**

The key of a given field’s key-value pair.

Version history:

- 2.4.0: added

value: **str**

The value associated with the *name* key.

Version history:

- 2.4.0: added

verified_at: **str | None**

Timestamp of when the server verified a URL value for a rel=”me” link. (nullable)

Version history:

- 2.6.0: added

class mastodon.return_types.**Role**(***kwargs*)

A role granting a user a set of permissions.

Example:

```
# Returns a Role object
mastodon.account_verify_credentials().role
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Role/

id: str | int | *MaybeSnowflakeIdType* | datetime

The ID of the Role in the database.

Version history:

- 4.0.0: added

name: str

The name of the role.

Version history:

- 4.0.0: added

permissions: str

A bitmask that represents the sum of all permissions granted to the role.

Version history:

- 4.0.0: added

color: str

The hex code assigned to this role. If no hex code is assigned, the string will be empty.

Version history:

- 4.0.0: added

highlighted: bool

Whether the role is publicly visible as a badge on user profiles.

Version history:

- 4.0.0: added

class mastodon.return_types.**CredentialAccountSource**(***kwargs*)

Source values useful for editing a user's profile.

Example:

```
# Returns a CredentialAccountSource object
mastodon.account_verify_credentials()["source"]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Account/

privacy: str

The user's default visibility setting ("private", "unlisted" or "public").

Version history:

- 1.5.0: added

sensitive: bool

Denotes whether user media should be marked sensitive by default.

Version history:

- 1.5.0: added

note: str

Plain text version of the user's bio.

Version history:

- 1.5.0: added

language: str | None

The default posting language for new statuses. (nullable) Should contain (as text): TwoLetterLanguageCodeEnum

Version history:

- 2.4.2: added

fields: *NonPaginatableList*[*AccountField*]

Metadata about the account.

Version history:

- 2.4.0: added

follow_requests_count: int

The number of pending follow requests.

Version history:

- 3.0.0: added

indexable: bool

Boolean indicating whether public posts by this user should be searchable by anyone.

Version history:

- 4.2.0: added

hide_collections: bool

Boolean indicating whether the user has chosen to hide their network (followers/followed accounts).

Version history:

- 4.1.0: added

discoverable: bool | None

Indicates whether or not the user is visible on the discovery page. (nullable)

Version history:

- 3.1.0: added

attribution_domains: *NonPaginatableList*[str]

List of domains that are allowed to be shown as having published something from this user.

Version history:

- 4.4.0: added

class mastodon.return_types.**Status**(***kwargs*)

A single status / toot / post.

Example:

```
# Returns a Status object
mastodon.toot("Hello from Python")
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Status/

id: *MaybeSnowflakeIdType*

Id of this status.

Version history:

- 0.1.0: added

uri: **str**

Descriptor for the status EG 'tag:mastodon.social,2016-11-25:objectId=<id>:objectType=Status'.

Version history:

- 0.1.0: added

url: **str | None**

URL of the status. (nullable) Should contain (as text): URL

Version history:

- 0.1.0: added

account: *Account*

Account which posted the status.

Version history:

- 0.1.0: added

in_reply_to_id: *MaybeSnowflakeIdType* **| None**

Id of the status this status is in response to. (nullable)

Version history:

- 0.1.0: added

in_reply_to_account_id: *MaybeSnowflakeIdType* **| None**

Id of the account this status is in response to. (nullable)

Version history:

- 0.1.0: added

reblog: *Status* **| None**

Denotes whether the status is a reblog. If so, set to the original status. (nullable)

Version history:

- 0.1.0: added

content: **str**

Content of the status, as HTML: ' Hello from Python '. Should contain (as text): HTML

Version history:

- 0.1.0: added

`created_at: datetime`

Creation time.

Version history:

- 0.1.0: added

`reblogs_count: int`

Number of reblogs.

Version history:

- 0.1.0: added

`favourites_count: int`

Number of favourites.

Version history:

- 0.1.0: added

`reblogged: bool | None`

Denotes whether the logged in user has boosted this status. (optional)

Version history:

- 0.1.0: added

`favourited: bool | None`

Denotes whether the logged in user has favourited this status. (optional)

Version history:

- 0.1.0: added

`sensitive: bool`

Denotes whether media attachments to the status are marked sensitive.

Version history:

- 0.9.9: added

`spoiler_text: str`

Warning text that should be displayed before the status content.

Version history:

- 1.0.0: added

`visibility: str`

Toot visibility. Should contain (as text): VisibilityEnum

Version history:

- 0.9.9: added

`mentions: NonPaginatableList[StatusMention]`

A list of StatusMention this status includes.

Version history:

- 0.6.0: added

media_attachments: *NonPaginatableList*[*MediaAttachment*]

List files attached to this status.

Version history:

- 0.6.0: added

emojis: *NonPaginatableList*[*CustomEmoji*]

A list of CustomEmoji used in the status.

Version history:

- 2.0.0: added

tags: *NonPaginatableList*[*Tag*]

A list of Tags used in the status.

Version history:

- 0.6.0: added

bookmarked: bool | None

True if the status is bookmarked by the logged in user, False if not. (optional)

Version history:

- 3.1.0: added

application: *Application* | None

Application for the client used to post the status (Does not federate and is therefore always None for remote statuses, can also be None for local statuses for some legacy applications registered before this field was introduced). (optional)

Version history:

- 0.9.9: added

language: str | None

The language of the status, if specified by the server, as ISO 639-1 (two-letter) language code. (nullable) Should contain (as text): TwoLetterLanguageCodeEnum

Version history:

- 1.4.0: added

muted: bool | None

Boolean denoting whether the user has muted this status by way of conversation muting. (optional)

Version history:

- 1.4.0: added

pinned: bool | None

Boolean denoting whether or not the status is currently pinned for the associated account. (optional)

Version history:

- 1.6.0: added

replies_count: int

The number of replies to this status.

Version history:

- 2.5.0: added

card: *PreviewCard* | **None**

A preview card for links from the status, if present at time of delivery. (nullable)

Version history:

- 2.6.0: added

poll: *Poll* | **None**

A poll object if a poll is attached to this status. (nullable)

Version history:

- 2.8.0: added

edited_at: **datetime** | **None**

Time the status was last edited. (nullable)

Version history:

- 3.5.0: added

filtered: *NonPaginatableList*[*FilterResult*] | **None**

If present, a list of filter application results that indicate which of the users filters matched and what actions should be taken. (optional)

Version history:

- 4.0.0: added

quote: *Quote* | *ShallowQuote* | **None**

Information about a quoted status. Can be shallow (ShallowQuote, id only) or full (Quote, full status object included). (nullable)

Version history:

- 4.4.0: added

class mastodon.return_types.**Quote**(***kwargs*)

A full quote of a status, including the full status object in case of an accepted quote None if the quote is not accepted.

Example:

```
# Returns a Quote object
mastodon.status(<status id>).quote
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Quote/

state: **str**

The state of the quote.

Version history:

- 4.4.0: added

quoted_status: *Status* | **None**

The quoted status object, if the quote has been accepted. (optional)

Version history:

- 4.4.0: added

class mastodon.return_types.**ShallowQuote**(***kwargs*)

A shallow quote of a status, containing only the ID of the quoted status. Used in multi-level quotes.

Example:

```
# Returns a ShallowQuote object
mastodon.status(<status id>).quote.quoted_status.quote
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/ShallowQuote/

quoted_status_id: *MaybeSnowflakeIdType* **| None**

The ID of the quoted status. None if the quote is not accepted. (nullable)

Version history:

- 4.4.0: added

state: **str**

The state of the quote.

Version history:

- 4.4.0: added

class mastodon.return_types.**StatusEdit**(***kwargs*)

An object representing a past version of an edited status.

Example:

```
# Returns a StatusEdit object
mastodon.status_history(<status id>)[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/StatusEdit/

content: **str**

Content for this version of the status.

Version history:

- 3.5.0: added

spoiler_text: **str**

CW / Spoiler text for this version of the status.

Version history:

- 3.5.0: added

sensitive: **bool**

Whether media in this version of the status is marked as sensitive.

Version history:

- 3.5.0: added

created_at: **datetime**

Time at which this version of the status was posted.

Version history:

- 3.5.0: added

account: *Account*

Account object of the user that posted the status.

Version history:

- 3.5.0: added

media_attachments: *NonPaginatableList*[*MediaAttachment*]

List of MediaAttachment objects with the attached media for this version of the status.

Version history:

- 3.5.0: added

emojis: *NonPaginatableList*[*CustomEmoji*]

List of custom emoji used in this version of the status.

Version history:

- 3.5.0: added

poll: *Poll* | **None**

The current state of the poll options at this revision. Note that edits changing the poll options will be collapsed together into one edit, since this action resets the poll. (optional)

Version history:

- 3.5.0: added

class mastodon.return_types.**FilterResult**(***kwargs*)

A filter action that should be taken on a status.

Example:

```
# Returns a FilterResult object
mastodon.status(<status id>).filtered[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/FilterResult/

filter: *Filter* | *FilterV2*

The filter that was matched.

Version history:

- 4.0.0: added

keyword_matches: *NonPaginatableList*[**str**] | **None**

The keyword within the filter that was matched. (nullable)

Version history:

- 4.0.0: added

status_matches: *NonPaginatableList* | **None**

The status ID within the filter that was matched. (nullable)

Version history:

- 4.0.0: added

class mastodon.return_types.**StatusMention**(***kwargs*)

A mention of a user within a status.

Example:

```
# Returns a StatusMention object
mastodon.toot("@admin he doing it sideways").mentions[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Mention/

url: str

Mentioned user's profile URL (potentially remote). Should contain (as text): URL

Version history:

- 0.6.0: added

username: str

Mentioned user's user name (not including domain).

Version history:

- 0.6.0: added

acct: str

Mentioned user's account name (including domain).

Version history:

- 0.6.0: added

id: str | int | *MaybeSnowflakeIdType* | datetime

Mentioned user's (local) account ID.

Version history:

- 0.6.0: added

class mastodon.return_types.**ScheduledStatus**(***kwargs*)

A scheduled status / toot to be eventually posted.

Example:

```
# Returns a ScheduledStatus object
mastodon.status_post("futureposting", scheduled_at=the_future)
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/ScheduledStatus/

id: str | int | *MaybeSnowflakeIdType* | datetime

Scheduled status ID (note: Not the id of the status once it gets posted!).

Version history:

- 2.7.0: added

scheduled_at: datetime

datetime object describing when the status is to be posted.

Version history:

- 2.7.0: added

params: *ScheduledStatusParams*

Parameters for the scheduled status, specifically.

Version history:

- 2.7.0: added

media_attachments: *NonPaginatableList*

Array of MediaAttachment objects for the attachments to the scheduled status.

Version history:

- 2.7.0: added

class mastodon.return_types.**ScheduledStatusParams**(***kwargs*)

Parameters for a status / toot to be posted in the future.

Example:

```
# Returns a ScheduledStatusParams object
mastodon.status_post("futureposting... 2", scheduled_at=the_future).params
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/ScheduledStatus/

text: **str**

Toot text.

Version history:

- 2.7.0: added

in_reply_to_id: *MaybeSnowflakeIdType* **| None**

ID of the status this one is a reply to. (nullable)

Version history:

- 2.7.0: added

media_ids: *NonPaginatableList*[**str**] **| None**

IDs of media attached to this status. (nullable)

Version history:

- 2.7.0: added

sensitive: **bool | None**

Whether this status is sensitive or not. (nullable)

Version history:

- 2.7.0: added

visibility: **str | None**

Visibility of the status. (nullable)

Version history:

- 2.7.0: added

idempotency: **str | None**

Idempotency key for the scheduled status. (nullable)

Version history:

- 2.7.0: added

scheduled_at: **datetime | None**

Present, but generally "None". Unsure what this is for - the actual scheduled_at is in the ScheduledStatus object, not here. If you know, let me know. (nullable)

Version history:

- 2.7.0: added

spoiler_text: str | None

CW text for this status. (nullable)

Version history:

- 2.7.0: added

application_id: str | int | *MaybeSnowflakeIdType* | datetime

ID of the application that scheduled the status.

Version history:

- 2.7.0: added

poll: *Poll* | None

Poll parameters. (nullable)

Version history:

- 2.8.0: added

language: str | None

The language that will be used for the status. (nullable) Should contain (as text): TwoLetterLanguageCodeEnum

Version history:

- 2.7.0: added

allowed_mentions: *NonPaginatableList*[str] | None

Undocumented. If you know what this does, please let me know. (nullable)

Version history:

- 2.7.0: added

with_rate_limit: bool

Whether the status should be rate limited. It is unclear to me what this does. If you know, please let met know.

Version history:

- 2.7.0: added

quoted_status_id: *MaybeSnowflakeIdType* | None

ID for a status this status will quote, once posted. (nullable)

Version history:

- 4.4.0: added

class mastodon.return_types.**Poll**(***kwargs*)

A poll attached to a status.

Example:

```
# Returns a Poll object
mastodon.poll(<poll id>)
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Poll/

id: `str | int | MaybeSnowflakeIdType | datetime`

The polls ID.

Version history:

- 2.8.0: added

expires_at: `datetime | None`

The time at which the poll is set to expire. (nullable)

Version history:

- 2.8.0: added

expired: `bool`

Boolean denoting whether users can still vote in this poll.

Version history:

- 2.8.0: added

multiple: `bool`

Boolean indicating whether it is allowed to vote for more than one option.

Version history:

- 2.8.0: added

votes_count: `int`

Total number of votes cast in this poll.

Version history:

- 2.8.0: added

voted: `bool`

Boolean indicating whether the logged-in user has already voted in this poll.

Version history:

- 2.8.0: added

options: *NonPaginatableList*[*PollOption*]

The poll options.

Version history:

- 2.8.0: added

emojis: *NonPaginatableList*[*CustomEmoji*]

List of CustomEmoji used in answer strings,.

Version history:

- 2.8.0: added

own_votes: *NonPaginatableList*[int]

The logged-in users votes, as a list of indices to the options.

Version history:

- 2.8.0: added

voters_count: int | None

How many unique accounts have voted on a multiple-choice poll. (nullable)

Version history:

- 2.8.0: added

class mastodon.return_types.**PollOption**(***kwargs*)

A poll option within a poll.

Example:

```
# Returns a PollOption object
mastodon.poll(<poll id>).options[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Poll/

title: str

Text of the option.

Version history:

- 2.8.0: added

votes_count: int | None

Count of votes for the option. Can be None if the poll creator has chosen to hide vote totals until the poll expires and it hasn't yet. (nullable)

Version history:

- 2.8.0: added

class mastodon.return_types.**Conversation**(***kwargs*)

A conversation (using direct / mentions-only visibility) between two or more users.

Example:

```
# Returns a Conversation object
mastodon.conversations()[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Conversation/

id: str | int | *MaybeSnowflakeIdType* | datetime

The ID of this conversation object.

Version history:

- 2.6.0: added

unread: bool

Boolean indicating whether this conversation has yet to be read by the user.

Version history:

- 2.6.0: added

accounts: *NonPaginatableList*[*Account*]

List of accounts (other than the logged-in account) that are part of this conversation.

Version history:

- 2.6.0: added

last_status: *Status* | **None**

The newest status in this conversation. (nullable)

Version history:

- 2.6.0: added

class mastodon.return_types.**Tag**(***kwargs*)

A hashtag, as part of a status or on its own (e.g. trending).

Example:

```
# Returns a Tag object
mastodon.trending_tags()[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Tag/

name: **str**

Hashtag name (not including the #).

Version history:

- 0.9.0: added

url: **str**

Hashtag URL (can be remote). Should contain (as text): URL

Version history:

- 0.9.0: added

history: *NonPaginatableList*[*TagHistory*] | **None**

List of TagHistory for up to 7 days. Not present in statuses. (optional)

Version history:

- 2.4.1: added

following: **bool** | **None**

Boolean indicating whether the logged-in user is following this tag. (optional)

Version history:

- 4.0.0: added

id: **str** | **None**

The ID of the Tag in the database. Only present for data returned from admin endpoints. (optional)

Version history:

- 3.5.0: added

trendable: **bool** | **None**

Whether the hashtag has been approved to trend. Only present for data returned from admin endpoints. (optional)

Version history:

- 3.5.0: added

usable: **bool** | **None**

Whether the hashtag has not been disabled from auto-linking. Only present for data returned from admin endpoints. (optional)

Version history:

- 3.5.0: added

requires_review: bool | None

Whether the hashtag has not been reviewed yet to approve or deny its trending. Only present for data returned from admin endpoints. (optional)

Version history:

- 3.5.0: added

featuring: bool | None

Whether the hashtag is featured on the logged-in users profile. (optional)

Version history:

- 4.4.0: added

class mastodon.return_types.**TagHistory**(***kwargs*)

Usage history for a hashtag.

Example:

```
# Returns a TagHistory object
mastodon.trending_tags()[0].history[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Tag/

day: datetime

Date of the day this TagHistory is for. Should contain (as text): datetime

Version history:

- 2.4.1: added

uses: str

Number of statuses using this hashtag on that day.

Version history:

- 2.4.1: added

accounts: str

Number of accounts using this hashtag in at least one status on that day.

Version history:

- 2.4.1: added

class mastodon.return_types.**CustomEmoji**(***kwargs*)

A custom emoji.

Example:

```
# Returns a CustomEmoji object
mastodon.toot(":sidekiqin:").emojis[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/CustomEmoji/

shortcode: str

Emoji shortcode, without surrounding colons.

Version history:

- 2.0.0: added

url: str

URL for the emoji image, can be animated. Should contain (as text): URL

Version history:

- 2.0.0: added

static_url: str

URL for the emoji image, never animated. Should contain (as text): URL

Version history:

- 2.0.0: added

visible_in_picker: bool

True if the emoji is enabled, False if not.

Version history:

- 2.0.0: added

category: str | None

The category to display the emoji under (not present if none is set). (nullable)

Version history:

- 3.0.0: added

class mastodon.return_types.**Application**(***kwargs*)

Information about an app (in terms of the API).

Example:

```
# Returns a Application object
mastodon.app_verify_credentials()
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Application/

id: str | int | *MaybeSnowflakeIdType* | datetime

ID of the application.

Version history:

- 2.7.0: added

name: str

The applications name.

Version history:

- 0.9.9: added

website: str | None

The applications website. (nullable)

Version history:

- 0.9.9: added

- 3.5.1: this property is now nullable

vapid_key: str

THIS FIELD IS DEPRECATED. IT IS RECOMMENDED THAT YOU DO NOT USE IT.

A vapid key that can be used in web applications.

Version history:

- 2.8.0: added
- 4.3.0: deprecated

redirect_uri: str

THIS FIELD IS DEPRECATED. IT IS RECOMMENDED THAT YOU DO NOT USE IT.

The applications redirect URI or urn:ietf:wg:oauth:2.0:oob. Deprecated, it is recommended to use redirect_uris instead.

Version history:

- 0.0.0: added
- 4.3.0: deprecated

redirect_uris: *NonPaginatableList*[str]

The applications redirect URI or urn:ietf:wg:oauth:2.0:oob. Deprecated, it is recommended to use redirect_uris instead. Should contain (as text): URL

Version history:

- 4.3.0: added

scopes: *NonPaginatableList*[str]

The applications available scopes. Should contain (as text): Scopes

Version history:

- 4.3.0: added

class mastodon.return_types.**Relationship**(***kwargs*)

Information about the relationship between two users.

Example:

```
# Returns a Relationship object
mastodon.account_relationships(<account id>)[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Relationship/

id: str | int | *MaybeSnowflakeIdType* | datetime

ID of the relationship object.

Version history:

- 0.6.0: added

following: bool

Boolean denoting whether the logged-in user follows the specified user.

Version history:

- 0.6.0: added

`followed_by: bool`

Boolean denoting whether the specified user follows the logged-in user.

Version history:

- 0.6.0: added

`blocking: bool`

Boolean denoting whether the logged-in user has blocked the specified user.

Version history:

- 0.6.0: added

`blocked_by: bool`

Boolean denoting whether the logged-in user has been blocked by the specified user, if information is available.

Version history:

- 2.8.0: added

`muting: bool`

Boolean denoting whether the logged-in user has muted the specified user.

Version history:

- 1.1.0: added

`muting_notifications: bool`

Boolean denoting wheter the logged-in user has muted notifications related to the specified user.

Version history:

- 2.1.0: added

`requested: bool`

Boolean denoting whether the logged-in user has sent the specified user a follow request.

Version history:

- 0.9.9: added

`domain_blocking: bool`

Boolean denoting whether the logged-in user has blocked the specified users domain.

Version history:

- 1.4.0: added

`showing_reblogs: bool`

Boolean denoting whether the specified users reblogs show up on the logged-in users Timeline.

Version history:

- 2.1.0: added

`endorsed: bool`

Boolean denoting wheter the specified user is being endorsed / featured by the logged-in user.

Version history:

- 2.5.0: added

note: str

A free text note the logged in user has created for this account (not publicly visible).

Version history:

- 3.2.0: added

notifying: bool

Boolean indicating whether the logged-in user has enabled notifications for this users posts.

Version history:

- 3.3.0: added

languages: *NonPaginatableList*[str] | None

List of languages that the logged in user is following this user for (if any). (nullable) Should contain (as text): TwoLetterLanguageCodeEnum

Version history:

- 4.0.0: added

requested_by: bool

Boolean indicating whether the specified user has sent the logged-in user a follow request.

Version history:

- 0.9.9: added

class mastodon.return_types.**FilterV2**(***kwargs*)

Information about a keyword / status filter.

Example:

```
# Returns a FilterV2 object
mastodon.filters_v2()[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Filter/

id: str | int | *MaybeSnowflakeIdType* | datetime

Id of the filter.

Version history:

- 4.0.0: added

context: *NonPaginatableList*[str]

List of places where the filters are applied. Should contain (as text): FilterContextEnum

Version history:

- 4.0.0: added

expires_at: datetime | None

Expiry date for the filter. (nullable)

Version history:

- 4.0.0: added

title: str

Name the user has chosen for this filter.

Version history:

- 4.0.0: added

filter_action: str

The action to be taken when a status matches this filter. Should contain (as text): FilterActionEnum

Version history:

- 4.0.0: added

keywords: *NonPaginatableList*[*FilterKeyword*]

A list of keywords that will trigger this filter.

Version history:

- 4.0.0: added

statuses: *NonPaginatableList*[*FilterStatus*]

A list of statuses that will trigger this filter.

Version history:

- 4.0.0: added

class mastodon.return_types.**Notification**(***kwargs*)

A notification about some event, like a new reply or follower.

Example:

```
# Returns a Notification object
mastodon.notifications()[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Notification/

id: str | int | *MaybeSnowflakeIdType* | datetime

id of the notification.

Version history:

- 0.9.9: added

type: str

"mention", "reblog", "favourite", "follow", "poll" or "follow_request". Should contain (as text): NotificationTypeEnum

Version history:

- 0.9.9: added
- 2.8.0: added *poll*
- 3.1.0: added *follow_request*
- 3.3.0: added *status*
- 3.5.0: added *update* and *admin.sign_up*
- 4.0.0: added *admin.report*
- 4.3.0: added *severed_relationships* and *moderation_warning*

created_at: datetime

The time the notification was created.

Version history:

- 0.9.9: added

account: *Account*

Account of the user from whom the notification originates.

Version history:

- 0.9.9: added

status: *Status* | **None**

In case of "mention", the mentioning status In case of reblog / favourite, the reblogged / favourited status. (optional)

Version history:

- 0.9.9: added
- 4.0.0: is now optional

group_key: **str**

A key to group notifications by. Structure is unspecified and subject to change, so please do not make assumptions about it.

Version history:

- 4.3.0: added

report: *Report* | **None**

Report that was the object of the notification. (optional)

Version history:

- 4.0.0: added

event: *RelationshipSeveranceEvent* | **None**

Summary of the event that caused follow relationships to be severed. (optional)

Version history:

- 4.3.0: added

moderation_warning: *AccountWarning* | **None**

Moderation warning that caused the notification. (optional)

Version history:

- 4.3.0: added

class mastodon.return_types.**Context**(***kwargs*)

The conversation context for a given status, i.e. its predecessors (that it replies to) and successors (that reply to it).

Example:

```
# Returns a Context object
mastodon.status_context(<status id>)
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Context/

ancestors: *NonPaginatableList*[*Status*]

A list of Statuses that the Status with this Context is a reply to.

Version history:

- 0.6.0: added

descendants: *NonPaginatableList*[*Status*]

A list of Statuses that are replies to the Status with this Context.

Version history:

- 0.6.0: added

class mastodon.return_types.**UserList**(***kwargs*)

A list of users.

Example:

```
# Returns a UserList object
mastodon.lists()[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/List/

id: str | int | *MaybeSnowflakeIdType* | datetime

id of the list.

Version history:

- 2.1.0: added

title: str

title of the list.

Version history:

- 2.1.0: added

replies_policy: str

Which replies should be shown in the list. Should contain (as text): RepliesPolicyEnum

Version history:

- 3.3.0: added

exclusive: bool | None

Boolean indicating whether users on this list are removed from the home feed (appearing exclusively as part of the list). nb: This setting applies to posts at the time they are put into a feed. (optional)

Version history:

- 4.2.0: added

class mastodon.return_types.**MediaAttachment**(***kwargs*)

A piece of media (like an image, video, or audio file) that can be or has been attached to a status.

Example:

```
# Returns a MediaAttachment object
mastodon.media_post("image.jpg", "image/jpeg")["meta"]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/MediaAttachment/

id: *MaybeSnowflakeIdType*

The ID of the attachment.

Version history:

- 0.6.0: added

type: str

Media type: 'image', 'video', 'gifv', 'audio' or 'unknown'.

Version history:

- 0.6.0: added
- 2.9.1: added *audio*

url: str

The URL for the image in the local cache. Should contain (as text): URL

Version history:

- 0.6.0: added

remote_url: str | None

The remote URL for the media (if the image is from a remote instance). (nullable) Should contain (as text): URL

Version history:

- 0.6.0: added

preview_url: str | None

The URL for the media preview. (nullable) Should contain (as text): URL

Version history:

- 0.6.0: added

text_url: str | None

THIS FIELD IS DEPRECATED. IT IS RECOMMENDED THAT YOU DO NOT USE IT.

Deprecated. The display text for the media (what shows up in text). May not be present in mastodon versions after 3.5.0. (optional) Should contain (as text): URL

Version history:

- 0.6.0: added
- 3.5.0: removed

meta: *MediaAttachmentMetadataContainer*

MediaAttachmentMetadataContainer that contains metadata for 'original' and 'small' (preview) versions of the MediaAttachment. Either may be empty. May additionally contain an "fps" field giving a videos frames per second (possibly rounded), and a "length" field giving a videos length in a human-readable format. Note that a video may have an image as preview. May also contain a 'focus' object and a media 'colors' object.

Version history:

- 1.5.0: added
- 2.3.0: added focus
- 4.0.0: added colors

blurhash: str

The blurhash for the image, used for preview / placeholder generation. Should contain (as text): Blurhash

Version history:

- 2.8.1: added

description: str | None

If set, the user-provided description for this media. (nullable)

Version history:

- 2.0.0: added

preview_remote_url: str | None

If set, the remote URL for the thumbnail of this media attachment on the or originating instance. (nullable) Should contain (as text): URL

Version history:

- 0.6.0: added

class mastodon.return_types.**MediaAttachmentMetadataContainer**(***kwargs*)

An object holding metadata about a media attachment and its thumbnail. In addition to the documented fields, there may be additional fields. These are not documented, not guaranteed to be present (they are a Mastodon implementation detail), and may change without notice, so relying on them is not recommended.

Example:

```
# Returns a MediaAttachmentMetadataContainer object
mastodon.media_post("audio.mp3").meta
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/MediaAttachment/

original: *MediaAttachmentImageMetadata* | *MediaAttachmentVideoMetadata* | *MediaAttachmentAudioMetadata*

Metadata for the original media attachment.

Version history:

- 0.6.0: added

small: *MediaAttachmentImageMetadata*

Metadata for the thumbnail of this media attachment.

Version history:

- 0.6.0: added

colors: *MediaAttachmentColors* | None

Information about accent colors for the media. (optional)

Version history:

- 4.0.0: added

focus: *MediaAttachmentFocusPoint* | None

Information about the focus point for the media. (optional)

Version history:

- 3.3.0: added

class mastodon.return_types.**MediaAttachmentImageMetadata**(***kwargs*)

Metadata for an image media attachment.

Example:

```
# Returns a MediaAttachmentImageMetadata object
mastodon.media_post("image.jpg").meta.original
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/MediaAttachment/

width: int

Width of the image in pixels.

Version history:

- 0.6.0: added

height: int

Height of the image in pixels.

Version history:

- 0.6.0: added

aspect: float

Aspect ratio of the image as a floating point number.

Version history:

- 0.6.0: added

size: str

Textual representation of the image size in pixels, e.g. '800x600'.

Version history:

- 0.6.0: added

class mastodon.return_types.**MediaAttachmentVideoMetadata**(***kwargs*)

Metadata for a video attachment. This can be a proper video, or a gifv (a looping, soundless animation). Both use the same data model currently, though there is a possibility that they could be split in the future.

Example:

```
# Returns a MediaAttachmentVideoMetadata object
mastodon.media_post("video.mp4").meta.original
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/MediaAttachment/

width: int

Width of the video in pixels.

Version history:

- 0.6.0: added

height: int

Height of the video in pixels.

Version history:

- 0.6.0: added

frame_rate: str

Exact frame rate of the video in frames per second. Can be an integer fraction (i.e. "20/7").

Version history:

- 0.6.0: added

duration: float

Duration of the video in seconds.

Version history:

- 0.6.0: added

bitrate: int

Average bit-rate of the video in bytes per second.

Version history:

- 0.6.0: added

class mastodon.return_types.**MediaAttachmentAudioMetadata**(***kwargs*)

Metadata for an audio media attachment.

Example:

```
# Returns a MediaAttachmentAudioMetadata object
mastodon.media_post("audio.mp3").meta.original
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/MediaAttachment/

duration: float

Duration of the audio file in seconds.

Version history:

- 0.6.0: added

bitrate: int

Average bit-rate of the audio file in bytes per second.

Version history:

- 0.6.0: added

class mastodon.return_types.**MediaAttachmentFocusPoint**(***kwargs*)

The focus point for a media attachment, for cropping purposes.

Example:

```
# Returns a MediaAttachmentFocusPoint object
mastodon.media_post("image.jpg").meta.focus
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/MediaAttachment/

x: float

Focus point x coordinate (between -1 and 1), with 0 being the center and -1 and 1 being the left and right edges respectively.

Version history:

- 2.3.0: added

y: float

Focus point x coordinate (between -1 and 1), with 0 being the center and -1 and 1 being the upper and lower edges respectively.

Version history:

- 2.3.0: added

class mastodon.return_types.**MediaAttachmentColors**(***kwargs*)

Object describing the accent colors for a media attachment.

Example:

```
# Returns a MediaAttachmentColors object
mastodon.media_post("image.jpg").meta.colors
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/MediaAttachment/

foreground: str

Estimated foreground colour for the attachment thumbnail, as a html format hex color (#rrggbb).

Version history:

- 4.0.0: added

background: str

Estimated background colour for the attachment thumbnail, as a html format hex color (#rrggbb).

Version history:

- 4.0.0: added

accent: str

Estimated accent colour for the attachment thumbnail.

Version history:

- 4.0.0: added

class mastodon.return_types.**PreviewCard**(***kwargs*)

A preview card attached to a status, e.g. for an embedded video or link.

Example:

```
# Returns a PreviewCard object
mastodon.status_card(<status id>)
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/PreviewCard/

url: str

The URL of the card. Should contain (as text): URL

Version history:

- 1.0.0: added

title: str

The title of the card.

Version history:

- 1.0.0: added

description: str

Description of the embedded content.

Version history:

- 1.0.0: added

`type: str`

Embed type: 'link', 'photo', 'video', or 'rich'.

Version history:

- 1.3.0: added

`image: str | None`

(optional) The image associated with the card. (nullable) Should contain (as text): URL

Version history:

- 1.0.0: added

`author_name: str`

THIS FIELD IS DEPRECATED. IT IS RECOMMENDED THAT YOU DO NOT USE IT.

Name of the embedded contents author. Deprecated in favour of the *authors* field.

Version history:

- 1.3.0: added
- 4.3.0: deprecated

`author_url: str`

URL pointing to the embedded contents author. Deprecated in favour of the *authors* field. Should contain (as text): URL

Version history:

- 1.3.0: added
- 4.3.0: deprecated

`width: int`

Width of the embedded object.

Version history:

- 1.3.0: added

`height: int`

Height of the embedded object.

Version history:

- 1.3.0: added

`html: str`

HTML string representing the embed. Should contain (as text): HTML

Version history:

- 1.3.0: added

`provider_name: str`

Name of the provider from which the embed originates.

Version history:

- 1.3.0: added

provider_url: str

URL pointing to the embeds provider. Should contain (as text): URL

Version history:

- 1.3.0: added

blurhash: str | None

Blurhash of the preview image. (nullable) Should contain (as text): Blurhash

Version history:

- 3.2.0: added

language: str | None

Language of the embedded content. (optional) Should contain (as text): TwoLetterLanguageCodeEnum

Version history:

- 1.3.0: added

embed_url: str

Used for photo embeds, instead of custom *html*. Should contain (as text): URL

Version history:

- 2.1.0: added

authors: *NonPaginatableList*[*PreviewCardAuthor*]

List of fediverse accounts of the authors of this post, as *PreviewCardAuthor*.

Version history:

- 4.3.0: added

image_description: str

Alt text / image description for the image preview for the card.

Version history:

- 4.2.0: added

published_at: datetime | None

Publication time of the embedded content, if available, as a *datetime* object. (nullable)

Version history:

- 4.2.0: added

history: *NonPaginatableList*[*TrendingLinkHistory*] | None

Only present for trending links. A list of TrendingLinkHistory objects. (optional)

Version history:

- 3.5.0: added

class mastodon.return_types.**TrendingLinkHistory**(***kwargs*)

A history entry for a trending link.

Example:

```
# Returns a TrendingLinkHistory object
mastodon.trending_links()[0].history[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/PreviewCard/#trends-link

day: datetime

The day this history entry is for, as a *datetime* object.

Version history:

- 3.5.0: added

uses: int

The number of times this link was used on this day.

Version history:

- 3.5.0: added

accounts: int

The number of accounts that used this link on this day.

Version history:

- 3.5.0: added

class mastodon.return_types.**PreviewCardAuthor**(***kwargs*)

A preview card attached to a status, e.g. for an embedded video or link.

Example:

```
# Returns a PreviewCardAuthor object
mastodon.status_card(<status id>).authors[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/PreviewCardAuthor/

name: str

THIS FIELD IS DEPRECATED. IT IS RECOMMENDED THAT YOU DO NOT USE IT.

Name of the embedded contents author.

Version history:

- 4.3.0: added

url: str

URL pointing to the embedded contents author. Should contain (as text): URL

Version history:

- 4.3.0: added

account: *Account*

Account of the author of this post, as *Account*.

Version history:

- 4.3.0: added

class mastodon.return_types.**SearchV2**(***kwargs*)

A search result, with accounts, hashtags and statuses.

Example:

```
# Returns a SearchV2 object
mastodon.search("<search query>")
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Search/

accounts: *NonPaginatableList*[*Account*]

List of Accounts resulting from the query.

Version history:

- 1.1.0: added

hashtags: *NonPaginatableList*[*Tag*]

List of Tags resulting from the query.

Version history:

- 2.4.1: added

statuses: *NonPaginatableList*[*Status*]

List of Statuses resulting from the query.

Version history:

- 1.1.0: added

class mastodon.return_types.**Instance**(***kwargs*)

Information about an instance. V1 API version.

Example:

```
# Returns a Instance object
mastodon.instance_v1()
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/V1_Instance/

uri: str

The instance's domain name. Moved to 'domain' for the v2 API, though Mastodon.py will mirror it here for backwards compatibility. Should contain (as text): DomainName

Version history:

- 1.1.0: added

title: str

The instance's title.

Version history:

- 1.1.0: added

short_description: str

An very brief text only instance description. Moved to 'description' for the v2 API.

Version history:

- 2.9.2: added

description: str

THIS FIELD IS DEPRECATED. IT IS RECOMMENDED THAT YOU DO NOT USE IT.

A brief instance description set by the admin. The V1 variant could contain html, but this is now deprecated. Likely to be empty on many instances. Should contain (as text): HTML

Version history:

- 1.1.0: added
- 4.0.0: deprecated - likely to be empty.

email: str

The admin contact email. Moved to InstanceContacts for the v2 API, though Mastodon.py will mirror it here for backwards compatibility. Should contain (as text): Email

Version history:

- 1.1.0: added

version: str

The instance's Mastodon version. For a more robust parsed major/minor/patch version see TODO IMPLEMENT FUNCTION TO RETURN VERSIONS.

Version history:

- 1.3.0: added

urls: *InstanceURLs*

Additional InstanceURLs, in the v1 api version likely to be just 'streaming_api' with the stream server websocket address.

Version history:

- 1.4.2: added

stats: *InstanceStatistics* | None

InstanceStatistics containing three stats, user_count (number of local users), status_count (number of local statuses) and domain_count (number of known instance domains other than this one). This information is not present in the v2 API variant in this form - there is a 'usage' field instead. (optional)

Version history:

- 1.6.0: added

thumbnail: str | None

Information about thumbnails to represent the instance. In the v1 API variant, simply an URL pointing to a banner image representing the instance. The v2 API provides a more complex structure with a list of thumbnails of different sizes in this field. (nullable) Should contain (as text): URL

Version history:

- 1.6.1: added

languages: *NonPaginatableList*[str]

Array of ISO 639-1 (two-letter) language codes the instance has chosen to advertise. Should contain (as text): TwoLetterLanguageCodeEnum

Version history:

- 2.3.0: added

registrations: bool

A boolean indication whether registrations on this instance are open (True) or not (False). The v2 API variant instead provides a dict with more information about possible registration requirements here.

Version history:

- 1.6.0: added

approval_required: bool

True if account approval is required when registering, False if not. Moved to InstanceRegistrations object for the v2 API.

Version history:

- 2.9.2: added

invites_enabled: bool

THIS FIELD IS DEPRECATED. IT IS RECOMMENDED THAT YOU DO NOT USE IT.

Boolean indicating whether invites are enabled on this instance. Changed in 4.0.0 from being true when the instance setting to enable invites is true to be true when the default user role has invites enabled (i.e. everyone can invite people). The v2 API does not contain this field, and it is not clear whether it will stay around.

Version history:

- 3.1.4: added
- 4.0.0: changed specifics of when field is true, deprecated

configuration: *InstanceConfiguration*

Various instance configuration settings - especially various limits (character counts, media upload sizes, ...).

Version history:

- 3.1.4: added

contact_account: *Account*

Account of the primary contact for the instance. Moved to InstanceContacts for the v2 API.

Version history:

- 1.1.0: added

rules: *NonPaginatableList*[*Rule*]

List of Rules with *id* and *text* fields, one for each server rule set by the admin.

Version history:

- 3.4.0: added

class mastodon.return_types.**InstanceConfiguration**(***kwargs*)

Configuration values for this instance, especially limits and enabled features.

Example:

```
# Returns a InstanceConfiguration object
mastodon.instance_v1().configuration
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/methods/instance/

accounts: *InstanceAccountConfiguration*

Account-related instance configuration fields.

Version history:

- 3.4.2: added

statuses: *InstanceStatusConfiguration*

Status-related instance configuration fields.

Version history:

- 3.4.2: added

media_attachments: *InstanceMediaConfiguration*

Media-related instance configuration fields.

Version history:

- 3.4.2: added

polls: *InstancePollConfiguration*

Poll-related instance configuration fields.

Version history:

- 3.4.2: added

class mastodon.return_types.**InstanceURLs**(***kwargs*)

A list of URLs related to an instance.

Example:

```
# Returns a InstanceURLs object
mastodon.instance_v1().urls
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/V1_Instance/

streaming_api: **str**

The Websockets URL for connecting to the streaming API. Renamed to 'streaming' for the v2 API. Should contain (as text): URL

Version history:

- 3.4.2: added

class mastodon.return_types.**InstanceV2**(***kwargs*)

Information about an instance.

Example:

```
# Returns a InstanceV2 object
mastodon.instance_v2()
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Instance/

domain: **str**

The instances domain name. Should contain (as text): DomainName

Version history:

- 4.0.0: added

title: **str**

The instance's title.

Version history:

- 4.0.0: added

version: **str**

The instance's Mastodon version. For a more robust parsed major/minor/patch version see TODO IMPLEMENT FUNCTION TO RETURN VERSIONS.

Version history:

- 4.0.0: added

source_url: str

URL pointing to a copy of the source code that is used to run this instance. For Mastodon instances, the AGPL requires that this code be available. Should contain (as text): URL

Version history:

- 4.0.0: added

description: str

A brief instance description set by the admin. Contains what in the v1 version was the short description. Should contain (as text): HTML

Version history:

- 4.0.0: added

usage: *InstanceUsage*

Information about recent activity on this instance.

Version history:

- 4.0.0: added

thumbnail: *InstanceThumbnail* | None

Information about thumbnails to represent the instance. (nullable)

Version history:

- 4.0.0: added

languages: *NonPaginatableList*[str]

Array of ISO 639-1 (two-letter) language codes the instance has chosen to advertise. Should contain (as text): TwoLetterLanguageCodeEnum

Version history:

- 4.0.0: added

configuration: *InstanceConfigurationV2*

Various instance configuration settings - especially various limits (character counts, media upload sizes, ...).

Version history:

- 4.0.0: added

registrations: *InstanceRegistrations*

InstanceRegistrations object with information about how users can sign up on this instance.

Version history:

- 4.0.0: added

contact: *InstanceContact*

Contact information for this instance.

Version history:

- 4.0.0: added

rules: *NonPaginatableList*[*Rule*]

List of Rules with *id* and *text* fields, one for each server rule set by the admin.

Version history:

- 4.0.0: added

icon: *NonPaginatableList*[*InstanceIcon*]

The instance icon, as a list of *InstanceIcon* , with entries representing different available size variants. Should contain (as text): URL

Version history:

- 4.3.0: added

api_versions: *AttribAccessDict*

A list of API versions supported by this instance, each as an entry in a dict with the name of the implementation as the key (such as 'mastodon'). The exact format is unspecified, any fork or implementation can put what if feels like there. Mastodon currently puts just '2'.

Version history:

- 4.3.0: added

class mastodon.return_types.**InstanceIcon**(***kwargs*)

Icon for the instance, in a specific size.

Example:

```
# Returns a InstanceIcon object
mastodon.instance_v2().icon[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/methods/instance/#InstanceIcon

src: str

URL for this icon size. Should contain (as text): URL

Version history:

- 4.3.0: added

size: str

Textual representation of the icon size in pixels as (width)x(height) string, e.g. '64x64'.

Version history:

- 4.3.0: added

class mastodon.return_types.**InstanceConfigurationV2**(***kwargs*)

Configuration values for this instance, especially limits and enabled features.

Example:

```
# Returns a InstanceConfigurationV2 object
mastodon.instance_v2().configuration
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/methods/instance/

accounts: *InstanceAccountConfiguration*

Account-related instance configuration fields.

Version history:

- 3.4.2: added

statuses: *InstanceStatusConfiguration*

Status-related instance configuration fields.

Version history:

- 3.4.2: added

media_attachments: *InstanceMediaConfiguration*

Media-related instance configuration fields.

Version history:

- 3.4.2: added

polls: *InstancePollConfiguration*

Poll-related instance configuration fields.

Version history:

- 3.4.2: added

translation: *InstanceTranslationConfiguration*

Translation-related instance configuration fields. Only present for the v2 API variant of the instance API.

Version history:

- 4.0.0: added

urls: *InstanceURLsV2*

Instance related URLs. Only present for the v2 API variant of the instance API.

Version history:

- 4.0.0: added

vapid: *InstanceVapidKey*

VAPID key used by this instance to sign webpush requests. Only present for the v2 API variant of the instance API.

Version history:

- 4.3.0: added

limited_federation: **bool**

Whether federation on this instance is limited to explicitly allowed domains ('allowlist mode').

Version history:

- 4.4.0: added

class mastodon.return_types.**InstanceVapidKey**(***kwargs*)

The VAPID key used by this instance to sign webpush requests.

Example:

```
# Returns a InstanceVapidKey object
mastodon.instance_v2().configuration.vapid
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/methods/instance/

public_key: **str**

The public key in VAPID format.

Version history:

- 4.3.0: added

class mastodon.return_types.**InstanceURLsV2**(***kwargs*)

A list of URLs related to an instance.

Example:

```
# Returns a InstanceURLsV2 object
mastodon.instance_v2().configuration.urls
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Instance/

streaming: str

The Websockets URL for connecting to the streaming API. Should contain (as text): URL

Version history:

- 4.0.0: added

status: str | None

If present, a URL where the status and possibly current issues with the instance can be checked. (optional) Should contain (as text): URL

Version history:

- 4.0.0: added

about: str | None

If present, a URL where the instance's about page can be found. (optional) Should contain (as text): URL

Version history:

- 4.4.0: added

privacy_policy: str | None

If present, a URL where the instance's privacy policy can be found. (optional) Should contain (as text): URL

Version history:

- 4.4.0: added

terms_of_service: str | None

If present, a URL where the instance's terms of service can be found. (optional) Should contain (as text): URL

Version history:

- 4.4.0: added

class mastodon.return_types.**InstanceThumbnail**(***kwargs*)

Extended information about an instances thumbnail.

Example:

```
# Returns a InstanceThumbnail object
mastodon.instance().thumbnail
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/V1_Instance/

url: str

The URL for an image representing the instance. Should contain (as text): URL

Version history:

- 4.0.0: added

blurhash: str | None

The blurhash for the image representing the instance. (optional) Should contain (as text): Blurhash

Version history:

- 4.0.0: added

versions: *InstanceThumbnailVersions* | None

Different resolution versions of the image representing the instance. (optional)

Version history:

- 4.0.0: added

class mastodon.return_types.**InstanceThumbnailVersions**(***kwargs*)

Different resolution versions of the image representing the instance.

Example:

```
# Returns a InstanceThumbnailVersions object
mastodon.instance().thumbnail.versions
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Instance/

at1x: str | None

The URL for an image representing the instance, for devices with 1x resolution / 96 dpi. (optional) Should contain (as text): URL

Version history:

- 4.0.0: added

at2x: str | None

The URL for the image representing the instance, for devices with 2x resolution / 192 dpi. (optional) Should contain (as text): URL

Version history:

- 4.0.0: added

class mastodon.return_types.**InstanceStatistics**(***kwargs*)

Usage statistics for an instance.

Example:

```
# Returns a InstanceStatistics object
mastodon.instance_v1().stats
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Instance/

user_count: int

The total number of accounts that have been created on this instance.

Version history:

- 1.6.0: added

status_count: *int*

The total number of local posts that have been made on this instance.

Version history:

- 1.6.0: added

domain_count: *int*

The total number of other instances that this instance is aware of.

Version history:

- 1.6.0: added

class mastodon.return_types.**InstanceUsage**(***kwargs*)

Usage / recent activity information for this instance.

Example:

```
# Returns a InstanceUsage object
mastodon.instance().usage
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Instance/

users: *InstanceUsageUsers*

Information about user counts on this instance.

Version history:

- 3.0.0: added

class mastodon.return_types.**InstanceUsageUsers**(***kwargs*)

Recent active user information about this instance.

Example:

```
# Returns a InstanceUsageUsers object
mastodon.instance().usage.users
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Instance/

active_month: *int*

This instances most recent monthly active user count.

Version history:

- 3.0.0: added

class mastodon.return_types.**RuleTranslation**(***kwargs*)

A translation for a rule into a specific language.

Example:

```
# Returns a RuleTranslation object
mastodon.instance().rules[0].translations['de']
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Rule/#translations

text: *str*

The rule to be followed, in few words, in the specified language.

Version history:

- 4.4.0: added

hint: str

Potentially, the rule to be followed, in more words, in the specified language.

Version history:

- 4.4.0: added

class mastodon.return_types.**Rule**(***kwargs*)

A rule that instance staff has specified users must follow on this instance.

Example:

```
# Returns a Rule object
mastodon.instance().rules[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Rule/

id: str | int | *MaybeSnowflakeIdType* | datetime

An identifier for the rule.

Version history:

- 3.4.0: added

text: str

The rule to be followed, in few words.

Version history:

- 3.4.0: added

hint: str

Potentially, the rule to be followed, in more words.

Version history:

- 4.3.0: added

translations: *AttribAccessDict*[str, *RuleTranslation*]

A list of translations for the rule, as a dictionary with the key being ISO 639-1 (two-letter) language codes for available languages.

Version history:

- 4.4.0: added

class mastodon.return_types.**InstanceRegistrations**(***kwargs*)

Registration information for this instance, like whether registrations are open and whether they require approval.

Example:

```
# Returns a InstanceRegistrations object
mastodon.instance_v2().registrations
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Instance/

approval_required: bool

Boolean indicating whether registrations on the instance require approval.

Version history:

- 4.0.0: added

enabled: bool

Boolean indicating whether registrations are enabled on this instance.

Version history:

- 4.0.0: added

message: str | None

A message to be shown instead of the sign-up form when registrations are closed. (nullable) Should contain (as text): HTML

Version history:

- 4.0.0: added

url: str | None

A custom URL for account registration, when using external authentication. (nullable) Should contain (as text): URL

Version history:

- 4.2.0: added

sign_up_url: str | None

URL to the sign-up form for this instance. Only present for the v2 API variant of the instance API. (optional)

Version history:

- 4.2.0: added

reason_required: bool | None

Boolean indicating whether a reason for registration is required on this instance. (nullable)

Version history:

- 4.4.0: added

min_age: int | None

Minimum age in years required to register on this instance. (nullable)

Version history:

- 4.4.0: added

class mastodon.return_types.**InstanceContact**(***kwargs*)

Contact information for this instances' staff.

Example:

```
# Returns a InstanceContact object
mastodon.instance().contact
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Instance/

account: *Account*

Account that has been designated as the instances contact account.

Version history:

- 4.0.0: added

email: str

E-mail address that can be used to contact the instance staff. Should contain (as text): Email

Version history:

- 4.0.0: added

class mastodon.return_types.**InstanceAccountConfiguration**(***kwargs*)

Configuration values relating to accounts.

Example:

```
# Returns a InstanceAccountConfiguration object
mastodon.instance().configuration.accounts
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/methods/instance/

max_featured_tags: int

The maximum number of featured tags that can be displayed on a profile.

Version history:

- 4.0.0: added

max_pinned_statuses: int

The maximum number of pinned statuses for an account.

Version history:

- 4.3.0: added

class mastodon.return_types.**InstanceStatusConfiguration**(***kwargs*)

Configuration values relating to statuses.

Example:

```
# Returns a InstanceStatusConfiguration object
mastodon.instance().configuration.statuses
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/methods/instance/

max_characters: int

Maximum number of characters in a status this instance allows local users to use.

Version history:

- 3.4.2: added

max_media_attachments: int

Maximum number of media attachments per status this instance allows local users to use.

Version history:

- 3.4.2: added

characters_reserved_per_url: int

Number of characters that this instance counts a URL as when counting charaters.

Version history:

- 3.4.2: added

class mastodon.return_types.**InstanceTranslationConfiguration**(***kwargs*)

Configuration values relating to translation.

Example:

```
# Returns a InstanceTranslationConfiguration object
mastodon.instance_v2().configuration.translation
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/methods/instance/

enabled: bool

Boolean indicating whether the translation API is enabled on this instance.

Version history:

- 4.0.0: added

class mastodon.return_types.**InstanceMediaConfiguration**(***kwargs*)

Configuration values relating to media attachments.

Example:

```
# Returns a InstanceMediaConfiguration object
mastodon.instance().configuration.media_attachments
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/methods/instance/

supported_mime_types: *NonPaginatableList*[str]

Mime types the instance accepts for media attachment uploads.

Version history:

- 3.4.2: added

image_size_limit: int

Maximum size (in bytes) the instance will accept for image uploads.

Version history:

- 3.4.2: added

image_matrix_limit: int

Maximum total number of pixels (i.e. width * height) the instance will accept for image uploads.

Version history:

- 3.4.2: added

video_size_limit: int

Maximum size (in bytes) the instance will accept for video uploads.

Version history:

- 3.4.2: added

video_frame_rate_limit: int

Maximum frame rate the instance will accept for video uploads.

Version history:

- 3.4.2: added

video_matrix_limit: int

Maximum total number of pixels (i.e. width * height) the instance will accept for video uploads.

Version history:

- 3.4.2: added

description_limit: int

Maximum number of characters in a media attachment description this instance allows local users to use.

Version history:

- 4.4.0: added

class mastodon.return_types.**InstancePollConfiguration**(***kwargs*)

Configuration values relating to polls.

Example:

```
# Returns a InstancePollConfiguration object
mastodon.instance().configuration.polls
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/methods/instance/

max_options: int

How many poll options this instance allows local users to use per poll.

Version history:

- 3.4.2: added

max_characters_per_option: int

Maximum number of characters this instance allows local users to use per poll option.

Version history:

- 3.4.2: added

min_expiration: int

The shortest allowed duration for a poll on this instance, in seconds.

Version history:

- 3.4.2: added

max_expiration: int

The longest allowed duration for a poll on this instance, in seconds.

Version history:

- 3.4.2: added

class mastodon.return_types.**Nodeinfo**(***kwargs*)

The instances standardized NodeInfo data.

Example:

```
# Returns a Nodeinfo object
mastodon.instance_nodeinfo()
```

See also (Mastodon API documentation): https://github.com/jhass/nodeinfo

version: str

Version of the nodeinfo schema spec that was used for this response.

Version history:

- 3.0.0: added

software: *NodeinfoSoftware*

Information about the server software being used on this instance.

Version history:

- 3.0.0: added

protocols: *NonPaginatableList*[str]

A list of strings specifying the federation protocols this instance supports. Typically, just "activitypub".

Version history:

- 3.0.0: added

services: *NodeinfoServices*

Services that this instance can retrieve messages from or send messages to.

Version history:

- 3.0.0: added

usage: *NodeinfoUsage*

Information about recent activity on this instance.

Version history:

- 3.0.0: added

openRegistrations: bool

Bool indicating whether the instance is open for registrations.

Version history:

- 3.0.0: added

metadata: *NodeinfoMetadata*

Additional node metadata. Can be entirely empty.

Version history:

- 3.0.0: added

class mastodon.return_types.**NodeinfoSoftware**(***kwargs*)

NodeInfo software-related information.

Example:

```
# Returns a NodeinfoSoftware object
mastodon.instance_nodeinfo().software
```

See also (Mastodon API documentation): https://github.com/jhass/nodeinfo

name: str

Name of the software used by this instance.

Version history:

- 3.0.0: added

version: str

String indicating the version of the software used by this instance.

Version history:

- 3.0.0: added

class mastodon.return_types.**NodeinfoServices**(***kwargs*)

Nodeinfo services-related information.

Example:

```
# Returns a NodeinfoServices object
mastodon.instance_nodeinfo().services
```

See also (Mastodon API documentation): https://github.com/jhass/nodeinfo

outbound: *NonPaginatableList*

List of services that this instance can send messages to. On Mastodon, typically an empty list.

Version history:

- 3.0.0: added

inbound: *NonPaginatableList*

List of services that this instance can retrieve messages from. On Mastodon, typically an empty list.

Version history:

- 3.0.0: added

class mastodon.return_types.**NodeinfoUsage**(***kwargs*)

Nodeinfo usage-related information.

Example:

```
# Returns a NodeinfoUsage object
mastodon.instance_nodeinfo().usage
```

See also (Mastodon API documentation): https://github.com/jhass/nodeinfo

users: *NodeinfoUsageUsers*

Information about user counts on this instance.

Version history:

- 3.0.0: added

localPosts: int

The total number of local posts that have been made on this instance.

Version history:

- 3.0.0: added

class mastodon.return_types.**NodeinfoUsageUsers**(***kwargs*)

Nodeinfo user count statistics.

Example:

```
# Returns a NodeinfoUsageUsers object
mastodon.instance_nodeinfo().usage.users
```

See also (Mastodon API documentation): https://github.com/jhass/nodeinfo

total: int

The total number of accounts that have been created on this instance.

Version history:

- 3.0.0: added

activeMonth: int

Number of users that have been active, by some definition (Mastodon: Have logged in at least once) in the last month.

Version history:

- 3.0.0: added

activeHalfyear: int

Number of users that have been active, by some definition (Mastodon: Have logged in at least once) in the last half year.

Version history:

- 3.0.0: added

class mastodon.return_types.**NodeinfoMetadata**(***kwargs*)

Nodeinfo extra metadata. Entirely freeform, be careful about consuming it programatically. Survey of real world usage: https://codeberg.org/thefederationinfo/nodeinfo_metadata_survey.

Example:

```
# Returns a NodeinfoMetadata object
mastodon.instance_nodeinfo().metadata
```

See also (Mastodon API documentation): https://github.com/jhass/nodeinfo

nodeName: str

Name of the instance, as specified by the instance admin.

Version history:

- 4.4.0: added

nodeDescription: str | None

Description of the instance, as specified by the instance admin. (nullable)

Version history:

- 4.4.0: added

class mastodon.return_types.**Activity**(***kwargs*)

Information about recent activity on an instance.

Example:

```
# Returns a Activity object
mastodon.instance_activity()[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/methods/instance/#activity

week: datetime

Date of the first day of the week the stats were collected for.

Version history:

- 2.1.2: added

logins: int

Number of users that logged in that week.

Version history:

- 2.1.2: added

registrations: int

Number of new users that week.

Version history:

- 2.1.2: added

statuses: int

Number of statuses posted that week.

Version history:

- 2.1.2: added

class mastodon.return_types.**Report**(***kwargs*)

Information about a report that has been filed against a user. Currently largely pointless, as updated reports cannot be fetched.

Example:

```
# Returns a Report object
mastodon.report(<account id>)
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Report/

id: str | int | *MaybeSnowflakeIdType* | datetime

Id of the report.

Version history:

- 2.9.1: added

action_taken: bool

True if a moderator or admin has processed the report, False otherwise.

Version history:

- 2.9.1: added

comment: str

Text comment submitted with the report.

Version history:

- 2.9.1: added

created_at: datetime

Time at which this report was created, as a datetime object.

Version history:

- 2.9.1: added

target_account: *Account*

Account that has been reported with this report.

Version history:

- 2.9.1: added

status_ids: *NonPaginatableList*[str | int | *MaybeSnowflakeIdType* | datetime]

List of status IDs attached to the report.

Version history:

- 2.9.1: added

action_taken_at: datetime | None

When an action was taken, if this report is currently resolved. (nullable)

Version history:

- 2.9.1: added

category: str

The category under which the report is classified. Should contain (as text): ReportReasonEnum

Version history:

- 3.5.0: added

forwarded: bool

Whether a report was forwarded to a remote instance.

Version history:

- 4.0.0: added

rules_ids: *NonPaginatableList*[str | int | *MaybeSnowflakeIdType* | datetime]

IDs of the rules selected for this report.

Version history:

- 3.5.0: added

class mastodon.return_types.**AdminReport**(***kwargs*)

Information about a report that has been filed against a user.

Example:

```
# Returns a AdminReport object
mastodon.admin_reports()[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Admin_Report/

id: str | int | *MaybeSnowflakeIdType* | datetime

Id of the report.

Version history:

- 2.9.1: added

action_taken: bool

True if a moderator or admin has processed the report, False otherwise.

Version history:

- 2.9.1: added

comment: str

Text comment submitted with the report.

Version history:

- 2.9.1: added

created_at: datetime

Time at which this report was created, as a datetime object.

Version history:

- 2.9.1: added

updated_at: datetime

Last time this report has been updated, as a datetime object.

Version history:

- 2.9.1: added

account: *Account*

Account of the user that filed this report.

Version history:

- 2.9.1: added

target_account: *Account*

Account that has been reported with this report.

Version history:

- 2.9.1: added

assigned_account: *AdminAccount* | None

If the report as been assigned to an account, that Account (None if not). (nullable)

Version history:

- 2.9.1: added

action_taken_by_account: *AdminAccount* | None

Account that processed this report. (nullable)

Version history:

- 2.9.1: added

statuses: *NonPaginatableList*[*Status*]

List of Statuses attached to the report.

Version history:

- 2.9.1: added

action_taken_at: `datetime | None`

When an action was taken, if this report is currently resolved. (nullable)

Version history:

- 2.9.1: added

category: `str`

The category under which the report is classified. Should contain (as text): ReportReasonEnum

Version history:

- 3.5.0: added

forwarded: `bool | None`

Whether a report was forwarded to a remote instance. Can be None. (nullable)

Version history:

- 4.0.0: added

rules: *NonPaginatableList*[*Rule*]

Rules attached to the report, for context.

Version history:

- 3.5.0: added

class mastodon.return_types.**WebPushSubscription**(***kwargs*)

Information about the logged-in users web push subscription for the authenticated application.

Example:

```
# Returns a WebPushSubscription object
mastodon.push_subscription()
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/WebPushSubscription/

id: `str | int |` *MaybeSnowflakeIdType* `| datetime`

Id of the push subscription.

Version history:

- 2.4.0: added

endpoint: `str`

Endpoint URL for the subscription. Should contain (as text): URL

Version history:

- 2.4.0: added

server_key: `str`

Server pubkey used for signature verification.

Version history:

- 2.4.0: added

alerts: *WebPushSubscriptionAlerts*

Subscribed events - object that may contain various keys, with value True if webpushes have been requested for those events.

Version history:

- 2.4.0: added
- 2.8.0: added poll`
- 3.1.0: added follow_request`
- 3.3.0: added status
- 3.5.0: added update and admin.sign_up
- 4.0.0: added admin.report

policy: str

Which sources should generate webpushes.

Version history:

- 2.4.0: added

standard: bool

Boolean indicatign whether the push messages follow the standardized specifications (RFC8030+RFC8291+RFC8292). Else they follow a legacy version of the specifications (4th draft of RFC8291 and 1st draft of RFC8292).

Version history:

- 4.4.0: added

class mastodon.return_types.**WebPushSubscriptionAlerts**(***kwargs*)

Information about alerts as part of a push subscription.

Example:

```
# Returns a WebPushSubscriptionAlerts object
mastodon.push_subscription().alerts
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/WebPushSubscription/

follow: bool | None

True if push subscriptions for follow events have been requested, false or not present otherwise. (nullable)

Version history:

- 2.4.0: added

favourite: bool | None

True if push subscriptions for favourite events have been requested, false or not present otherwise. (nullable)

Version history:

- 2.4.0: added

reblog: bool | None

True if push subscriptions for reblog events have been requested, false or not present otherwise. (nullable)

Version history:

- 2.4.0: added

mention: bool | None

True if push subscriptions for mention events have been requested, false or not present otherwise. (nullable)

Version history:

- 2.4.0: added

poll: bool | None

True if push subscriptions for poll events have been requested, false or not present otherwise. (nullable)

Version history:

- 2.8.0: added

follow_request: bool | None

True if push subscriptions for follow request events have been requested, false or not present otherwise. (nullable)

Version history:

- 2.4.0: added

status: bool | None

True if push subscriptions for status creation (watched users only) events have been requested, false or not present otherwise. (nullable)

Version history:

- 3.1.0: added

admin_sign_up: bool | None

True if push subscriptions for sign up events have been requested, false or not present otherwise. Admins only. (nullable)

Version history:

- 3.5.0: added

admin_report: bool | None

True if push subscriptions for report creation events have been requested, false or not present otherwise. Admins only. (nullable)

Version history:

- 4.0.0: added

class mastodon.return_types.**PushNotification**(***kwargs*)

A single Mastodon push notification received via WebPush, after decryption.

Example:

```
# Returns a PushNotification object
mastodon.push_subscription_decrypt_push(...)
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/WebPushSubscription/

access_token: str

Access token that can be used to access the API as the notified user.

Version history:

- 2.4.0: added

body: str

Text body of the notification.

Version history:

- 2.4.0: added

icon: str

URL to an icon for the notification. Should contain (as text): URL

Version history:

- 2.4.0: added

notification_id: str | int | *MaybeSnowflakeIdType* | datetime

ID that can be passed to notification() to get the full notification object,.

Version history:

- 2.4.0: added

notification_type: str

String indicating the type of notification.

Version history:

- 2.4.0: added

preferred_locale: str

The user's preferred locale. Should contain (as text): TwoLetterLanguageCodeEnum

Version history:

- 2.4.0: added

title: str

Title for the notification.

Version history:

- 2.4.0: added

class mastodon.return_types.**Preferences**(***kwargs*)

The logged in users preferences.

Example:

```
# Returns a Preferences object
mastodon.preferences()
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Preferences/

posting_default_visibility: str

Default visibility for new posts. Also found in CredentialAccountSource as *privacy*.

Version history:

- 2.8.0: added

posting_default_sensitive: bool

Default sensitivity flag for new posts. Also found in CredentialAccountSource as *sensitive*.

Version history:

- 2.8.0: added

posting_default_language: str | None

Default language for new posts. Also found in CredentialAccountSource as *language*. (nullable) Should contain (as text): TwoLetterLanguageCodeEnum

Version history:

- 2.8.0: added

reading_expand_media: str

String indicating whether media attachments should be automatically displayed or blurred/hidden.

Version history:

- 2.8.0: added

reading_expand_spoilers: bool

Boolean indicating whether CWs should be expanded by default.

Version history:

- 2.8.0: added

reading_autoplay_gifs: bool

Boolean indicating whether gifs should be autoplayed (True) or not (False).

Version history:

- 2.8.0: added

class mastodon.return_types.**FeaturedTag**(***kwargs*)

A tag featured on a users profile.

Example:

```
# Returns a FeaturedTag object
mastodon.featured_tags()[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/FeaturedTag/

id: str | int | *MaybeSnowflakeIdType* | datetime

The featured tags id.

Version history:

- 3.0.0: added

name: str

The featured tags name (without leading #).

Version history:

- 3.0.0: added

statuses_count: str

Number of publicly visible statuses posted with this hashtag that this instance knows about.

Version history:

- 3.0.0: added

last_status_at: datetime | None

The last time a public status containing this hashtag was added to this instance's database (can be None if there are none). (nullable)

Version history:

- 3.0.0: added

url: str

A link to all statuses by a user that contain this hashtag. Should contain (as text): URL

Version history:

- 3.3.0: added

class mastodon.return_types.**Marker**(***kwargs*)

A read marker indicating where the logged in user has left off reading a given timeline.

Example:

```
# Returns a Marker object
mastodon.markers_get()["home"]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Marker/

last_read_id: str | int | *MaybeSnowflakeIdType* | datetime

ID of the last read object in the timeline.

Version history:

- 3.0.0: added

version: int

A counter that is incremented whenever the marker is set to a new status.

Version history:

- 3.0.0: added

updated_at: datetime

The time the marker was last set, as a datetime object.

Version history:

- 3.0.0: added

class mastodon.return_types.**Announcement**(***kwargs*)

An announcement sent by the instances staff.

Example:

```
# Returns a Announcement object
mastodon.announcements()[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Announcement/

id: str | int | *MaybeSnowflakeIdType* | datetime

The annoucements id.

Version history:

- 3.1.0: added

content: str

The contents of the annoucement, as an html string. Should contain (as text): HTML

Version history:

- 3.1.0: added

starts_at: `datetime | None`

The annoucements start time, as a datetime object. Can be None. (nullable)

Version history:

- 3.1.0: added

ends_at: `datetime | None`

The annoucements end time, as a datetime object. Can be None. (nullable)

Version history:

- 3.1.0: added

all_day: `bool`

Boolean indicating whether the annoucement represents an "all day" event.

Version history:

- 3.1.0: added

published_at: `datetime`

The annoucements publish time, as a datetime object.

Version history:

- 3.1.0: added

updated_at: `datetime`

The annoucements last updated time, as a datetime object.

Version history:

- 3.1.0: added

read: `bool`

A boolean indicating whether the logged in user has dismissed the annoucement.

Version history:

- 3.1.0: added

mentions: *NonPaginatableList*[*StatusMention*]

Users mentioned in the annoucement.

Version history:

- 3.1.0: added

tags: *NonPaginatableList*

Hashtags mentioned in the announcement.

Version history:

- 3.1.0: added

emojis: *NonPaginatableList*

Custom emoji used in the annoucement.

Version history:

- 3.1.0: added

reactions: *NonPaginatableList*[*Reaction*]

Reactions to the annoucement.

Version history:

- 3.1.0: added

statuses: *NonPaginatableList*

Statuses linked in the announcement text.

Version history:

- 3.1.0: added

class mastodon.return_types.**Reaction**(***kwargs*)

A reaction to an announcement.

Example:

```
# Returns a Reaction object
mastodon.announcements()[0].reactions[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Reaction/

name: str

Name of the custom emoji or unicode emoji of the reaction.

Version history:

- 3.1.0: added

count: int

Reaction counter (i.e. number of users who have added this reaction).

Version history:

- 3.1.0: added

me: bool

True if the logged-in user has reacted with this emoji, false otherwise.

Version history:

- 3.1.0: added

url: str | None

URL for the custom emoji image. (nullable) Should contain (as text): URL

Version history:

- 3.1.0: added

static_url: str | None

URL for a never-animated version of the custom emoji image. (nullable) Should contain (as text): URL

Version history:

- 3.1.0: added

class mastodon.return_types.**StreamReaction**(***kwargs*)

A reaction to an announcement.

Example:

```
# Returns a StreamReaction object
# Only available via the streaming API
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/methods/streaming/

name: str

Name of the custom emoji or unicode emoji of the reaction.

Version history:

- 3.1.0: added

count: int

Reaction counter (i.e. number of users who have added this reaction).

Version history:

- 3.1.0: added

announcement_id: str | int | *MaybeSnowflakeIdType* | datetime

If of the announcement this reaction was for.

Version history:

- 3.1.0: added

class mastodon.return_types.**FamiliarFollowers**(***kwargs*)

A follower of a given account that is also followed by the logged-in user.

Example:

```
# Returns a FamiliarFollowers object
mastodon.account_familiar_followers(<account id>)[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/FamiliarFollowers/

id: str | int | *MaybeSnowflakeIdType* | datetime

ID of the account for which the familiar followers are being returned.

Version history:

- 3.5.0: added

accounts: *NonPaginatableList*[*Account*]

List of Accounts of the familiar followers.

Version history:

- 3.5.0: added

class mastodon.return_types.**AdminAccount**(***kwargs*)

Admin variant of the Account entity, with some additional information.

Example:

```
# Returns a AdminAccount object
mastodon.admin_account(<account id>)
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Admin_Account/

id: `str | int | MaybeSnowflakeIdType | datetime`

The users id,.

Version history:

- 2.9.1: added

username: `str`

The users username, no leading @.

Version history:

- 2.9.1: added

domain: `str | None`

The users domain. (nullable)

Version history:

- 2.9.1: added

created_at: `datetime`

The time of account creation.

Version history:

- 2.9.1: added

email: `str`

For local users, the user's email. Should contain (as text): Email

Version history:

- 2.9.1: added

ip: `str | None`

For local users, the user's last known IP address. (nullable)

Version history:

- 2.9.1: added
- 3.5.0: return type changed from String to AdminIP
- 4.0.0: bug fixed, return type is now a String again

role: *Role*

The users role.

Version history:

- 2.9.1: added, returns a String (enumerable, oneOf *user moderator admin*)
- 4.0.0: now uses Role entity

confirmed: `bool`

For local users, False if the user has not confirmed their email, True otherwise.

Version history:

- 2.9.1: added

suspended: bool

Boolean indicating whether the user has been suspended.

Version history:

- 2.9.1: added

silenced: bool

Boolean indicating whether the user has been silenced.

Version history:

- 2.9.1: added

disabled: bool

For local users, boolean indicating whether the user has had their login disabled.

Version history:

- 2.9.1: added

approved: bool

For local users, False if the user is pending, True otherwise.

Version history:

- 2.9.1: added

locale: str

For local users, the locale the user has set,. Should contain (as text): TwoLetterLanguageCodeEnum

Version history:

- 2.9.1: added

invite_request: str | None

If the user requested an invite, the invite request comment of that user. (nullable)

Version history:

- 2.9.1: added

account: *Account*

The user's Account.

Version history:

- 2.9.1: added

sensitized: bool

Boolean indicating whether the account has been marked as force-sensitive.

Version history:

- 2.9.1: added

ips: *NonPaginatableList*[*AdminIp*]

All known IP addresses associated with this account.

Version history:

- 3.5.0: added

created_by_application_id: `str | int | MaybeSnowflakeIdType | datetime | None`

Present if the user was created by an application and set to the application id. (optional)

Version history:

- 2.9.1: added

invited_by_account_id: `str | int | MaybeSnowflakeIdType | datetime | None`

Present if the user was created via invite and set to the inviting users id. (optional)

Version history:

- 2.9.1: added

class mastodon.return_types.**AdminIp**(***kwargs*)

An IP address used by some user or other instance, visible as part of some admin APIs.

Example:

```
# Returns a AdminIp object
mastodon.admin_account(<account id>).ips[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Admin_Ip/

ip: `str`

The IP address.

Version history:

- 3.5.0: added

used_at: `str`

The timestamp of when the IP address was last used for this account.

Version history:

- 3.5.0: added

class mastodon.return_types.**AdminMeasure**(***kwargs*)

A measurement, such as the number of active users, as returned by the admin reporting API.

Example:

```
# Returns a AdminMeasure object
mastodon.admin_measures(datetime.now() - timedelta(hours=24*5), datetime.now(),␣
↪interactions=True)[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Admin_Measure/

key: `str`

Name of the measure returned. Should contain (as text): AdminMeasureTypeEnum

Version history:

- 3.5.0: added

unit: `str | None`

Unit for the measure, if available. (nullable)

Version history:

- 3.5.0: added

total: str

Value of the measure returned.

Version history:

- 3.5.0: added

human_value: str | None

Human readable variant of the measure returned. (nullable)

Version history:

- 3.5.0: added

previous_total: str | None

Previous measurement period value of the measure returned, if available. (nullable)

Version history:

- 3.5.0: added

data: *NonPaginatableList*[*AdminMeasureData*]

A list of AdminMeasureData with the measure broken down by date.

Version history:

- 3.5.0: added

class mastodon.return_types.**AdminMeasureData**(***kwargs*)

A single row of data for an admin reporting api measurement.

Example:

```
# Returns a AdminMeasureData object
mastodon.admin_measures(datetime.now() - timedelta(hours=24*5), datetime.now(),␣
↪active_users=True)[0].data[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Admin_Measure/

date: datetime

Date for this row.

Version history:

- 3.5.0: added

value: int

Value of the measure for this row.

Version history:

- 3.5.0: added

class mastodon.return_types.**AdminDimension**(***kwargs*)

A qualitative measurement about the server, as returned by the admin reporting api.

Example:

```
# Returns a AdminDimension object
mastodon.admin_dimensions(datetime.now() - timedelta(hours=24*5), datetime.now(),␣
↪languages=True)[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Admin_Dimension/

key: str

Name of the dimension returned.

Version history:

- 3.5.0: added

data: *NonPaginatableList*[*AdminDimensionData*]

A list of data AdminDimensionData objects.

Version history:

- 3.5.0: added

class mastodon.return_types.**AdminDimensionData**(***kwargs*)

A single row of data for qualitative measurements about the server, as returned by the admin reporting api.

Example:

```
# Returns a AdminDimensionData object
mastodon.admin_dimensions(datetime.now() - timedelta(hours=24*5), datetime.now(),␣
↪languages=True)[0].data[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Admin_Dimension/

key: str

category for this row.

Version history:

- 3.5.0: added

human_key: str

Human readable name for the category for this row, when available.

Version history:

- 3.5.0: added

value: int

Numeric value for the category.

Version history:

- 3.5.0: added

class mastodon.return_types.**AdminRetention**(***kwargs*)

User retention data for a given cohort, as returned by the admin reporting api.

Example:

```
# Returns a AdminRetention object
mastodon.admin_retention(datetime.now() - timedelta(hours=24*5), datetime.now())[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Admin_Cohort/

period: datetime

Starting time of the period that the data is being returned for.

Version history:

- 3.5.0: added

frequency: str

Time resolution (day or month) for the returned data.

Version history:

- 3.5.0: added

data: *NonPaginatableList*[*AdminCohort*]

List of AdminCohort objects.

Version history:

- 3.5.0: added

class mastodon.return_types.**AdminCohort**(***kwargs*)

A single data point regarding user retention for a given cohort, as returned by the admin reporting api.

Example:

```
# Returns a AdminCohort object
mastodon.admin_retention(datetime.now() - timedelta(hours=24*5), datetime.now())[0].
↪data[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Admin_Cohort/

date: datetime

Date for this entry.

Version history:

- 3.5.0: added

rate: float

Fraction of users retained.

Version history:

- 3.5.0: added

value: int

Absolute number of users retained.

Version history:

- 3.5.0: added

class mastodon.return_types.**AdminDomainBlock**(***kwargs*)

A domain block, as returned by the admin API.

Example:

```
# Returns a AdminDomainBlock object
mastodon.admin_domain_blocks()[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Admin_DomainBlock/

id: str | int | *MaybeSnowflakeIdType* | datetime

The ID of the DomainBlock in the database.

Version history:

- 4.0.0: added

`domain: str`

The domain that is not allowed to federate.

Version history:

- 4.0.0: added

`created_at: datetime`

When the domain was blocked from federating.

Version history:

- 4.0.0: added

`severity: str`

The policy to be applied by this domain block. Should contain (as text): AdminDomainLimitEnum

Version history:

- 4.0.0: added

`reject_media: bool`

Whether to reject media attachments from this domain.

Version history:

- 4.0.0: added

`reject_reports: bool`

Whether to reject reports from this domain.

Version history:

- 4.0.0: added

`private_comment: str | None`

A private comment (visible only to other moderators) for the domain block. (nullable)

Version history:

- 4.0.0: added

`public_comment: str | None`

A public comment (visible to either all users, or the whole world) for the domain block. (nullable)

Version history:

- 4.0.0: added

`obfuscate: bool`

Whether to obfuscate public displays of this domain block.

Version history:

- 4.0.0: added

`digest: str | None`

SHA256 hex digest of the blocked domain. (nullable)

Version history:

- 4.3.0: added

class mastodon.return_types.**AdminCanonicalEmailBlock**(***kwargs*)

An e-mail block that has been set up to prevent certain e-mails to be used when signing up, via hash matching.

Example:

```
# Returns a AdminCanonicalEmailBlock object
mastodon.admin_create_canonical_email_block(email=<some email>)
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Admin_CanonicalEmailBlock

id: str | int | *MaybeSnowflakeIdType* | datetime

The ID of the email block in the database.

Version history:

- 4.0.0: added

canonical_email_hash: str

The SHA256 hash of the canonical email address.

Version history:

- 4.0.0: added

class mastodon.return_types.**AdminDomainAllow**(***kwargs*)

The opposite of a domain block, specifically allowing a domain to federate when the instance is in allowlist mode.

Example:

```
# Returns a AdminDomainAllow object
mastodon.admin_domain_allows()[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Admin_DomainAllow

id: str | int | *MaybeSnowflakeIdType* | datetime

The ID of the DomainAllow in the database.

Version history:

- 4.0.0: added

domain: str

The domain that is allowed to federate.

Version history:

- 4.0.0: added

created_at: datetime

When the domain was allowed to federate.

Version history:

- 4.0.0: added

class mastodon.return_types.**AdminEmailDomainBlock**(***kwargs*)

A block that has been set up to prevent e-mails from certain domains to be used when signing up.

Example:

```
# Returns a AdminEmailDomainBlock object
mastodo.admin_email_domain_blocks()[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Admin_EmailDomainBlock

id: str | int | *MaybeSnowflakeIdType* | datetime

The ID of the EmailDomainBlock in the database.

Version history:

- 4.0.0: added

domain: str

The email domain that is not allowed to be used for signups.

Version history:

- 4.0.0: added

created_at: datetime

When the email domain was disallowed from signups.

Version history:

- 4.0.0: added

history: *NonPaginatableList*[*AdminEmailDomainBlockHistory*]

Usage statistics for given days (typically the past week).

Version history:

- 4.0.0: added

class mastodon.return_types.**AdminEmailDomainBlockHistory**(***kwargs*)

Historic data about attempted signups using e-mails from a given domain.

Example:

```
# Returns a AdminEmailDomainBlockHistory object
mastodo.admin_email_domain_blocks()[0].history[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Admin_EmailDomainBlock

day: datetime

The time (in day increments) for which this row of historical data is valid.

Version history:

- 4.0.0: added

accounts: int

The number of different account creation attempts that have been made.

Version history:

- 4.0.0: added

uses: int

The number of different ips used in account creation attempts.

Version history:

- 4.0.0: added

class mastodon.return_types.**AdminIpBlock**(***kwargs*)

An admin IP block, to prevent certain IP addresses or address ranges from accessing the instance.

Example:

```
# Returns a AdminIpBlock object
mastodon.admin_ip_blocks()[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Admin_IpBlock

id: str | int | *MaybeSnowflakeIdType* | datetime

The ID of the DomainBlock in the database.

Version history:

- 4.0.0: added

ip: str

The IP address range that is not allowed to federate.

Version history:

- 4.0.0: added

severity: str

The associated policy with this IP block.

Version history:

- 4.0.0: added

comment: str

The recorded reason for this IP block.

Version history:

- 4.0.0: added

created_at: datetime

When the IP block was created.

Version history:

- 4.0.0: added

expires_at: datetime | None

When the IP block will expire. (nullable)

Version history:

- 4.0.0: added

class mastodon.return_types.**DomainBlock**(***kwargs*)

A domain block that has been implemented by instance staff, limiting the way posts from the blocked instance are handled.

Example:

```
# Returns a DomainBlock object
mastodon.instance_domain_blocks()[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/DomainBlock

domain: str

The domain which is blocked. This may be obfuscated or partially censored.

Version history:

- 4.0.0: added

digest: str

The SHA256 hash digest of the domain string.

Version history:

- 4.0.0: added

severity: str

The level to which the domain is blocked. Should contain (as text): DomainLimitEnum

Version history:

- 4.0.0: added

comment: str | None

An optional reason for the domain block. (nullable)

Version history:

- 4.0.0: added

class mastodon.return_types.**ExtendedDescription**(***kwargs*)

An extended instance description that can contain HTML.

Example:

```
# Returns a ExtendedDescription object
mastodon.instance_extended_description()
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/ExtendedDescription

updated_at: datetime

A timestamp of when the extended description was last updated.

Version history:

- 4.0.0: added

content: str

The rendered HTML content of the extended description. Should contain (as text): HTML

Version history:

- 4.0.0: added

class mastodon.return_types.**FilterKeyword**(***kwargs*)

A keyword that is being matched as part of a filter.

Example:

```
# Returns a FilterKeyword object
mastodon.filters_v2()[0].keywords[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/FilterKeyword

id: *str | int | MaybeSnowflakeIdType | datetime*

The ID of the FilterKeyword in the database.

Version history:

- 4.0.0: added

keyword: *str*

The phrase to be matched against.

Version history:

- 4.0.0: added

whole_word: *bool*

Should the filter consider word boundaries? See implementation guidelines for filters().

Version history:

- 4.0.0: added

class mastodon.return_types.**FilterStatus**(***kwargs*)

A single status that is being matched as part of a filter.

Example:

```
# Returns a FilterStatus object
mastodon.filter_statuses_v2(mastodon.filters_v2()[0])[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/FilterStatus

id: *str | int | MaybeSnowflakeIdType | datetime*

The ID of the FilterStatus in the database.

Version history:

- 4.0.0: added

status_id: *MaybeSnowflakeIdType*

The ID of the Status that will be filtered.

Version history:

- 4.0.0: added

class mastodon.return_types.**StatusSource**(***kwargs*)

The source data of a status, useful when editing a status.

Example:

```
# Returns a StatusSource object
mastodon.status_source(<status id>)
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/StatusSource

id: *str | int | MaybeSnowflakeIdType | datetime*

ID of the status in the database.

Version history:

- 3.5.0: added

text: str

The plain text used to compose the status.

Version history:

- 3.5.0: added

spoiler_text: str

The plain text used to compose the status's subject or content warning.

Version history:

- 3.5.0: added

class mastodon.return_types.**Suggestion**(***kwargs*)

A follow suggestion.

Example:

```
# Returns a Suggestion object
mastodon.suggestions_v2()[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Suggestion

source: str

THIS FIELD IS DEPRECATED. IT IS RECOMMENDED THAT YOU DO NOT USE IT.

The reason this account is being suggested.

Version history:

- 3.4.0: added
- 4.3.0: deprecated

sources: *NonPaginatableList*[str]

The reasons this account is being suggested. Should contain (as text): SuggestionSourceEnum

Version history:

- 4.3.0: added

account: *Account*

The account being recommended to follow.

Version history:

- 3.4.0: added

class mastodon.return_types.**Translation**(***kwargs*)

A translation of a status.

Example:

```
# Returns a Translation object
mastodon.status_translate(<status_id>, 'de')
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Translation

content: str

The translated text of the status.

Version history:

- 4.0.0: added

detected_source_language: str

The language of the source text, as auto-detected by the machine translation provider.

Version history:

- 4.0.0: added

provider: str

The service that provided the machine translation.

Version history:

- 4.0.0: added

class mastodon.return_types.**AccountCreationError**(***kwargs*)

An error response returned when creating an account fails.

Example:

```
# Returns a AccountCreationError object
mastodon.create_account('halcy', 'secret', 'invalid email lol', True, return_
↪detailed_error=True)[1]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/methods/accounts/#create

error: str

The error as a localized string.

Version history:

- 2.7.0: added

details: *AccountCreationErrorDetails*

A dictionary giving more details about what fields caused errors and in which ways.

Version history:

- 3.4.0: added

class mastodon.return_types.**AccountCreationErrorDetails**(***kwargs*)

An object containing detailed errors for different fields in the account creation attempt.

Example:

```
# Returns a AccountCreationErrorDetails object
mastodon.create_account('halcy', 'secret', 'invalid email lol', False, return_
↪detailed_error=True)[1].details
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/methods/accounts/#create

username: *NonPaginatableList*[*AccountCreationErrorDetailsField*] | None

An object giving more details about an error caused by the username. (optional)

Version history:

- 3.4.0: added

password: *NonPaginatableList*[*AccountCreationErrorDetailsField*] | None

An object giving more details about an error caused by the password. (optional)

Version history:

- 3.4.0: added

email: *NonPaginatableList*[*AccountCreationErrorDetailsField*] | **None**

An object giving more details about an error caused by the e-mail. (optional)

Version history:

- 3.4.0: added

agreement: *NonPaginatableList*[*AccountCreationErrorDetailsField*] | **None**

An object giving more details about an error caused by the usage policy agreement. (optional)

Version history:

- 3.4.0: added

locale: *NonPaginatableList*[*AccountCreationErrorDetailsField*] | **None**

An object giving more details about an error caused by the locale. (optional)

Version history:

- 3.4.0: added

reason: *NonPaginatableList*[*AccountCreationErrorDetailsField*] | **None**

An object giving more details about an error caused by the registration reason. (optional)

Version history:

- 3.4.0: added

date_of_birth: *NonPaginatableList*[*AccountCreationErrorDetailsField*] | **None**

An object giving more details about an error caused by the date of birth. (optional)

Version history:

- 4.4.0: added

class mastodon.return_types.**AccountCreationErrorDetailsField**(***kwargs*)

An object giving details about what specifically is wrong with a given field in an account registration attempt.

Example:

```
# Returns a AccountCreationErrorDetailsField object
mastodon.create_account('halcy', 'secret', 'invalid email lol', True, return_
↪detailed_error=True)[1].details.email[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/methods/accounts/#create

error: **str**

A machine readable string giving an error category.

Version history:

- 3.4.0: added

description: **str**

A description of the issue as a localized string.

Version history:

- 3.4.0: added

class mastodon.return_types.**NotificationPolicy**(***kwargs*)

Represents the notification filtering policy of the user.

Example:

```
# Returns a NotificationPolicy object
mastodon.notifications_policy()
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/NotificationPolicy

for_not_following: str

Whether to accept, filter or drop notifications from accounts the user is not following.

Version history:

- 4.3.0: added

for_not_followers: str

Whether to accept, filter or drop notifications from accounts that are not following the user.

Version history:

- 4.3.0: added

for_new_accounts: str

Whether to accept, filter or drop notifications from accounts created in the past 30 days.

Version history:

- 4.3.0: added

for_private_mentions: str

Whether to accept, filter or drop notifications from private mentions.

Version history:

- 4.3.0: added

for_limited_accounts: str

Whether to accept, filter or drop notifications from accounts that were limited by a moderator.

Version history:

- 4.3.0: added

summary: *NotificationPolicySummary*

A summary of the filtered notifications.

Version history:

- 4.3.0: added

class mastodon.return_types.**NotificationPolicySummary**(***kwargs*)

A summary of the filtered notifications.

Example:

```
# Returns a NotificationPolicySummary object
mastodon.notifications_policy().summary
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/NotificationPolicy

pending_requests_count: int

Number of different accounts from which the user has non-dismissed filtered notifications. Capped at 100.

Version history:

- 4.3.0: added

pending_notifications_count: int

Number of total non-dismissed filtered notifications. May be inaccurate.

Version history:

- 4.3.0: added

class mastodon.return_types.**RelationshipSeveranceEvent**(***kwargs*)

Summary of a moderation or block event that caused follow relationships to be severed.

Example:

```
# Returns a RelationshipSeveranceEvent object
# There isn't really a good way to get this manually - you get it if a moderation␣
↪takes action.
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/RelationshipSeveranceEvent

id: str

The ID of the relationship severance event in the database.

Version history:

- 4.3.0: added

type: str

Type of event. Should contain (as text): RelationshipSeveranceEventType

Version history:

- 4.3.0: added

purged: bool

Whether the list of severed relationships is unavailable because the data has been purged.

Version history:

- 4.3.0: added

target_name: str

Name of the target of the moderation/block event. This is either a domain name or a user handle, depending on the event type.

Version history:

- 4.3.0: added

followers_count: int

Number of followers that were removed as result of the event.

Version history:

- 4.3.0: added

following_count: int

Number of accounts the user stopped following as result of the event.

Version history:

- 4.3.0: added

created_at: datetime

When the event took place.

Version history:

- 4.3.0: added

class mastodon.return_types.**GroupedNotificationsResults**(***kwargs*)

Container for grouped notifications plus referenced accounts and statuses.

Example:

```
# Returns a GroupedNotificationsResults object
mastodon.grouped_notifications()
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/GroupedNotificationsResults

accounts: *NonPaginatableList*[*Account*]

Accounts referenced by grouped notifications.

Version history:

- 4.3.0: added

partial_accounts: *NonPaginatableList*[*PartialAccountWithAvatar*] | None

Partial accounts referenced by grouped notifications. Only returned with expand_accounts=partial_avatars. (optional)

Version history:

- 4.3.0: added

statuses: *NonPaginatableList*[*Status*]

Statuses referenced by grouped notifications.

Version history:

- 4.3.0: added

notification_groups: *NonPaginatableList*[*NotificationGroup*]

The grouped notifications themselves. Is actually in fact paginatable, but via the parent object.

Version history:

- 4.3.0: added

class mastodon.return_types.**PartialAccountWithAvatar**(***kwargs*)

A stripped-down version of Account, containing only what is necessary to display avatars and a few other fields.

Example:

```
# Returns a PartialAccountWithAvatar object
mastodon.grouped_notifications().partial_accounts[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/GroupedNotificationsResults

id: str

The account ID.

Version history:

- 4.3.0: added

acct: str

The Webfinger account URI.

Version history:

- 4.3.0: added

url: str

The location of the user's profile page.

Version history:

- 4.3.0: added

avatar: str

An image icon (avatar) shown in the profile.

Version history:

- 4.3.0: added

avatar_static: str

A static version of the avatar. May differ if the main avatar is animated.

Version history:

- 4.3.0: added

locked: bool

Whether the account manually approves follow requests.

Version history:

- 4.3.0: added

bot: bool

Indicates that the account may perform automated actions.

Version history:

- 4.3.0: added

class mastodon.return_types.**NotificationGroup**(***kwargs*)

A group of related notifications, plus metadata for pagination.

Example:

```
# Returns a NotificationGroup object
mastodon.grouped_notifications().notification_groups[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/GroupedNotificationsResults

group_key: str

Group key identifying the grouped notifications. Treated as opaque.

Version history:

- 4.3.0: added

notifications_count: int

Total number of individual notifications in this group.

Version history:

- 4.3.0: added

type: str

The type of event that resulted in the notifications. Should contain (as text): NotificationTypeEnum

Version history:

- 4.3.0: added

most_recent_notification_id: str

ID of the most recent notification in the group.

Version history:

- 4.3.0: added

page_min_id: str | None

ID of the oldest notification in this group within the current page. (optional)

Version history:

- 4.3.0: added

page_max_id: str | None

ID of the newest notification in this group within the current page. (optional)

Version history:

- 4.3.0: added

latest_page_notification_at: datetime | None

Date at which the most recent notification within this group (in the current page) was created. (optional)

Version history:

- 4.3.0: added

sample_account_ids: *NonPaginatableList*[str]

IDs of some of the accounts who most recently triggered notifications in this group.

Version history:

- 4.3.0: added

status_id: str | None

ID of the Status that was the object of the notification. (optional)

Version history:

- 4.3.0: added

report: *Report* | None

Report that was the object of the notification. (optional)

Version history:

- 4.3.0: added

event: *RelationshipSeveranceEvent* | **None**

Summary of the event that caused follow relationships to be severed. (optional)

Version history:

- 4.3.0: added

moderation_warning: *AccountWarning* | **None**

Moderation warning that caused the notification. (optional)

Version history:

- 4.3.0: added

class mastodon.return_types.**AccountWarning**(***kwargs*)

Moderation warning against a particular account.

Example:

```
# Returns a AccountWarning object
# There isn't really a good way to get this manually - you get it if a moderator␣
↪takes action.
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/AccountWarning

id: **str**

The ID of the account warning in the database.

Version history:

- 4.3.0: added

action: **str**

Action taken against the account.

Version history:

- 4.3.0: added

text: **str**

Message from the moderator to the target account.

Version history:

- 4.3.0: added

status_ids: *NonPaginatableList*[**str**] | **None**

List of status IDs relevant to the warning. May be null. (nullable)

Version history:

- 4.3.0: added

target_account: *Account*

Account against which a moderation decision has been taken.

Version history:

- 4.3.0: added

appeal: *Appeal* | **None**

Appeal submitted by the target account, if any. (nullable)

Version history:

- 4.3.0: added

created_at: datetime

When the event took place.

Version history:

- 4.3.0: added

class mastodon.return_types.**UnreadNotificationsCount**(***kwargs*)

Rhe (capped) number of unread notifications for the current user.

Example:

```
# Returns a UnreadNotificationsCount object
mastodon.notifications_unread_count()
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/methods/notifications/#unread-count

count: int

The capped number of unread notifications. The cap is not documented.

Version history:

- 4.3.0: added

class mastodon.return_types.**Appeal**(***kwargs*)

Appeal against a moderation action.

Example:

```
# Returns a Appeal object
# There isn't really a good way to get this manually - you get it if a moderator␣
↪takes action.
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Appeal/

text: str

Text of the appeal from the moderated account to the moderators..

Version history:

- 4.3.0: added

state: str

State of the appeal. Should contain (as text): AppealStateEnum

Version history:

- 4.3.0: added

class mastodon.return_types.**NotificationRequest**(***kwargs*)

Represents a group of filtered notifications from a specific user.

Example:

```
# Returns a NotificationRequest object
mastodon.notification_requests()[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/NotificationRequest

id: `str`

The ID of the notification request in the database.

Version history:

- 4.3.0: added

created_at: `datetime`

When the first filtered notification from that user was created.

Version history:

- 4.3.0: added

updated_at: `datetime`

When the notification request was last updated.

Version history:

- 4.3.0: added

account: *Account*

The account that performed the action that generated the filtered notifications.

Version history:

- 4.3.0: added

notifications_count: `str`

How many of this account's notifications were filtered.

Version history:

- 4.3.0: added

last_status: *Status* | `None`

Most recent status associated with a filtered notification from that account. (optional)

Version history:

- 4.3.0: added

class mastodon.return_types.**SupportedLocale**(***kwargs*)

A locale supported by the instance.

Example:

```
# Returns a SupportedLocale object
mastodon.instance_languages()[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Instance

code: `str`

The locale code.

Version history:

- 4.2.0: added

name: `str`

The name of the locale.

Version history:

- 4.2.0: added

class mastodon.return_types.**OAuthServerInfo**(***kwargs*)

Information about the OAuth authorization server.

Example:

```
# Returns a OAuthServerInfo object
mastodon.oauth_authorization_server_info()
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/methods/oauth/#authorization-server-metadata

issuer: str

The issuer of the OAuth server. Can be used to avoid accidentally getting replies from a wrong server by comparing it against the `iss` field. Not currently used by Mastodon. Should contain (as text): URL

Version history:

- 4.3.0: added

authorization_endpoint: str

The endpoint for authorization requests. Should contain (as text): URL

Version history:

- 4.3.0: added

token_endpoint: str

The endpoint for token requests. Should contain (as text): URL

Version history:

- 4.3.0: added

revocation_endpoint: str

The endpoint for revoking tokens. Should contain (as text): URL

Version history:

- 4.3.0: added

userinfo_endpoint: str

The endpoint for retrieving OAuth user information for the logged in user. Should contain (as text): URL

Version history:

- 4.4.0: added

scopes_supported: *NonPaginatableList*[str]

List of scopes supported by the OAuth server.

Version history:

- 4.3.0: added

response_types_supported: *NonPaginatableList*[str]

List of response types (i.e. what kind of parameters can the server get back to your callback) supported by the OAuth server.

Version history:

- 4.3.0: added

response_modes_supported: *NonPaginatableList*[str]

List of response modes (i.e. how does the server get callback parameters back to you) supported by the OAuth server.

Version history:

- 4.3.0: added

grant_types_supported: *NonPaginatableList*[str]

List of grant types (i.e. authorization methods) supported by the OAuth server.

Version history:

- 4.3.0: added

token_endpoint_auth_methods_supported: *NonPaginatableList*[str]

List of authentication methods supported by the token endpoint.

Version history:

- 4.3.0: added

code_challenge_methods_supported: *NonPaginatableList*[str]

List of code challenge methods supported by the OAuth server.

Version history:

- 4.3.0: added

service_documentation: str | None

URL to the service documentation (e.g. the Mastodon API reference). (optional)

Version history:

- 4.3.0: added

app_registration_endpoint: str | None

Endpoint for registering applications. (optional)

Version history:

- 4.3.0: added

class mastodon.return_types.**OAuthUserInfo**(***kwargs*)

Information about the currently logged in user, returned by the OAuth userinfo endpoint.

Example:

```
# Returns a OAuthUserInfo object
mastodon.oauth_userinfo()
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/methods/oauth/#userinfo

iss: str

The issuer of the OAuth server. Can be used to avoid accidentally getting replies from a wrong server by comparing it against the *issuer* field in OAuthServerInfo. Should contain (as text): URL

Version history:

- 4.4.0: added

sub: str

The subject identifier of the user. For Mastodon, the URI of the ActivityPub Actor document. Should contain (as text): URL

Version history:

- 4.4.0: added

name: str

The display name of the user.

Version history:

- 4.4.0: added

preferred_username: str

The preferred username of the user, i.e. the part after the first and before the second @ in their account name.

Version history:

- 4.4.0: added

profile: str

The URL of the user's profile page. Should contain (as text): URL

Version history:

- 4.4.0: added

picture: str

The URL of the user's profile picture. Should contain (as text): URL

Version history:

- 4.4.0: added

class mastodon.return_types.**TermsOfService**(***kwargs*)

The terms of service for the instance.

Example:

```
# Returns a TermsOfService object
mastodon.instance_terms_of_service()
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/methods/instance/#terms_of_service

effective_date: datetime

The date when the terms of service became effective.

Version history:

- 4.4.0: added

effective: bool

Whether the terms of service are currently in effect.

Version history:

- 4.4.0: added

content: str

The contents of the terms of service. Should contain (as text): HTML

Version history:

- 4.4.0: added

succeeded_by: datetime | None

If there are newer terms of service, their effective date. (optional)

Version history:

- 4.4.0: added

4.5 Deprecated types

class mastodon.return_types.**Filter**(***kwargs*)

Information about a keyword / status filter.

THIS ENTITY IS DEPRECATED. IT IS RECOMMENDED THAT YOU DO NOT USE IT.

Example:

```
# Returns a Filter object
mastodon.filters()[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/V1_Filter/

id: str | int | *MaybeSnowflakeIdType* | datetime

Id of the filter.

Version history:

- 2.4.3: added

phrase: str

Filtered keyword or phrase.

Version history:

- 2.4.3: added

context: *NonPaginatableList*[str]

List of places where the filters are applied. Should contain (as text): FilterContextEnum

Version history:

- 2.4.3: added
- 3.1.0: added *account*

expires_at: datetime | None

Expiry date for the filter. (nullable)

Version history:

- 2.4.3: added

irreversible: bool

Boolean denoting if this filter is executed server-side or if it should be ran client-side.

Version history:

- 2.4.3: added

whole_word: bool

Boolean denoting whether this filter can match partial words.

Version history:

- 2.4.3: added

class mastodon.return_types.**Search**(***kwargs*)

A search result, with accounts, hashtags and statuses.

THIS ENTITY IS DEPRECATED. IT IS RECOMMENDED THAT YOU DO NOT USE IT.

Example:

```
# Returns a Search object
mastodon.search_v1("<search query>")
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Search/

accounts: *NonPaginatableList*[*Account*]

List of Accounts resulting from the query.

Version history:

- 1.1.0: added

hashtags: *NonPaginatableList*[str]

THIS FIELD IS DEPRECATED. IT IS RECOMMENDED THAT YOU DO NOT USE IT.

List of Tags resulting from the query.

Version history:

- 1.1.0: added
- 2.4.1: v1 search deprecated because it returns a list of strings. v2 search added which returns a list of tags.
- 3.0.0: v1 removed

statuses: *NonPaginatableList*[*Status*]

List of Statuses resulting from the query.

Version history:

- 1.1.0: added

class mastodon.return_types.**IdentityProof**(***kwargs*)

A cryptographic proof-of-identity. Deprecated since 3.5.0.

THIS ENTITY IS DEPRECATED. IT IS RECOMMENDED THAT YOU DO NOT USE IT.

Example:

```
# Returns a IdentityProof object
# Deprecated since 3.5.0 and eventually removed, there is no way to get this on␣
↪current versions of Mastodon.
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/IdentityProof

provider: str

The name of the identity provider.

Version history:

- 2.8.0: added

provider_username: str

The account owner's username on the identity provider's service.

Version history:

- 2.8.0: added

updated_at: datetime

When the identity proof was last updated.

Version history:

- 2.8.0: added

proof_url: str

A link to a statement of identity proof, hosted by the identity provider.

Version history:

- 2.8.0: added

profile_url: str

The account owner's profile URL on the identity provider.

Version history:

- 2.8.0: added

4.6 Error handling

When Mastodon.py encounters an error, it will raise an exception, generally with some text included to tell you what went wrong.

The base class that all Mastodon exceptions inherit from is *MastodonError*. If you are only interested in the fact an error was raised somewhere in Mastodon.py, and not the details, this is the exception you can catch.

MastodonIllegalArgumentError is generally a programming problem - you asked the API to do something obviously invalid (i.e. specify a privacy option that does not exist).

MastodonFileNotFoundError and *MastodonNetworkError* are IO errors - could be you specified a wrong URL, could be the internet is down or your hard drive is dying. They inherit from *MastodonIOError*, for easy catching. There is a sub-error of *MastodonNetworkError*, *MastodonReadTimeout*, which is thrown when a streaming API stream times out during reading.

MastodonAPIError is an error returned from the Mastodon instance - the server has decided it can't fulfil your request (i.e. you requested info on a user that does not exist). It is further split into *MastodonNotFoundError* (API returned 404) and *MastodonUnauthorizedError* (API returned 401). Different error codes might exist, but are not currently handled separately.

MastodonMalformedEventError is raised when a streaming API listener receives an invalid event. There have been reports that this can sometimes happen after prolonged operation due to an upstream problem in the requests/urllib libraries.

MastodonRatelimitError is raised when you hit an API rate limit. You should try again after a while (see the rate limiting section above).

MastodonServerError is raised when the server throws an internal error, likely due to server misconfiguration.

MastodonVersionError is raised when a version check for an API call fails.

MastodonDeprecationWarning is raised when a deprecated API call is used. This is based on the *deprecation* HTTP header returned by the server.

4.7 App registration, authentication and preferences

Before you can use the Mastodon API, you have to register your application (which gets you a client key and client secret) and then log in (which gets you an access token) and out (revoking the access token you are logged in with). These functions allow you to do those things. Additionally, it is also possible to programmatically register a new user.

For convenience, once you have a client id, secret and access token, you can simply pass them to the constructor of the class, too!

Note that while it is perfectly reasonable to log back in whenever your app starts, registering a new application on every startup is not, so don't do that - instead, register an application once, and then persist your client id and secret. A convenient method for this is provided by the functions dealing with registering the app, logging in and the Mastodon classes constructor.

4.7.1 App registration and information

static Mastodon.**create_app**(*client_name*, *scopes: List[str] = ['read', 'write', 'follow', 'push']*, *redirect_uris: str | List[str] | None = None*, *website: str | None = None*, *to_file: str | PurePath | None = None*, *api_base_url: str | None = None*, *request_timeout: float = 300*, *session: Session | None = None*, *user_agent: str = 'mastodonpy'*) → Tuple[str, str]

Create a new app with given *client_name* and *scopes* (The basic scopes are "read", "write", "follow" and "push" - more granular scopes are available, please refer to Mastodon documentation for which) on the instance given by *api_base_url*. If you pass scopes, you must pass the same set of scopes to *log_in()* and *auth_request_url()*, otherwise, your auth request will fail.

Specify *redirect_uris* if you want users to be redirected to a certain page after authenticating in an OAuth flow. You can specify multiple URLs by passing a list. Note that if you wish to use OAuth authentication with redirects, the redirect URI must be one of the URLs specified here.

Specify *to_file* to persist your app's info to a file so you can use it in the constructor. Specify *website* to give a website for your app.

Specify *session* with a requests.Session for it to be used instead of the default. This can be used to, amongst other things, adjust proxy or SSL certificate settings.

Specify *user_agent* if you want to use a specific name as *User-Agent* header, otherwise "mastodonpy" will be used.

Presently, app registration is open by default, but this is not guaranteed to be the case for all Mastodon instances in the future.

Returns *client_id* and *client_secret*, both as strings.

Mastodon.**app_verify_credentials**() → *Application*

Fetch information about the current application.

Added: Mastodon v2.0.0, last changed: Mastodon v2.7.2 (parameters), Mastodon v4.3.0 (return value)

4.7.2 Authentication

Mastodon.**__init__**(*client_id: str | PurePath | None = None*, *client_secret: str | None = None*, *access_token: str | PurePath | None = None*, *api_base_url: str | None = None*, *debug_requests: bool = False*, *ratelimit_method: str = 'wait'*, *ratelimit_pacefactor: float = 1.1*, *request_timeout: float = 300*, *mastodon_version: str | None = None*, *version_check_mode: str = 'none'*, *session: Session | None = None*, *feature_set: str = 'mainline'*, *user_agent: str = 'mastodonpy'*, *lang: str | None = None*)

Create a new API wrapper instance based on the given *client_secret* and *client_id* on the instance given by *api_base_url*. If you give a *client_id* and it is not a file, you must also give a secret. If you specify an *access_token* then you don't need to specify a *client_id*. It is allowed to specify neither - in this case, you will be restricted to only using endpoints that do not require authentication. If a file is given as *client_id*, client ID, secret and base url are read from that file.

You can also specify an *access_token*, directly or as a file (as written by *log_in()*). If a file is given, Mastodon.py also tries to load the base URL from this file, if present. A client id and secret are not required in this case.

Mastodon.py can try to respect rate limits in several ways, controlled by *ratelimit_method*. "throw" makes functions throw a *MastodonRatelimitError* when the rate limit is hit. "wait" mode will, once the limit is hit, wait and retry the request as soon as the rate limit resets, until it succeeds. "pace" works like throw, but tries to wait in between calls so that the limit is generally not hit (how hard it tries to avoid hitting the rate limit can be controlled by ratelimit_pacefactor). The default setting is "wait". Note that even in "wait" and "pace" mode, requests can still fail due to network or other problems! Also note that "pace" and "wait" are NOT thread safe.

By default, a timeout of 300 seconds is used for all requests. If you wish to change this, pass the desired timeout (in seconds) as *request_timeout*.

For fine-tuned control over the requests object use *session* with a requests.Session.

The *mastodon_version* parameter can be used to specify the version of Mastodon that Mastodon.py will expect to be installed on the server. The function will throw an error if an unparseable Version is specified. If no version is specified, Mastodon.py will set *mastodon_version* to the detected version.

feature_set can be used to enable behaviour specific to non-mainline Mastodon API implementations. Details are documented in the functions that provide such functionality. Currently supported feature sets are *mainline*, *fedibird* and *pleroma*.

For some Mastodon instances a *User-Agent* header is needed. This can be set by parameter *user_agent*. Starting from Mastodon.py 1.5.2 *create_app()* stores the application name into the client secret file. If *client_id* points to this file, the app name will be used as *User-Agent* header as default. It is possible to modify old secret files and append a client app name to use it as a *User-Agent* name.

lang can be used to change the locale Mastodon will use to generate responses. Valid parameters are all ISO 639-1 (two letter) or for a language that has none, 639-3 (three letter) language codes. This affects some error messages (those related to validation) and trends. You can change the language using set_language().

The version check mode can be set to "none" (now the default behaviour), "changed" or "created". If set to "created", Mastodon.py will throw an error if the version of Mastodon it is connected to is too old to have an endpoint. If it is set to "changed", it will throw an error if the endpoint's behaviour has changed after the version of Mastodon that is connected has been released. If it is set to "none", version checking is disabled. When encountering problems, I would recommend setting this to "created" and/or setting *debug_requests* to True to get a better idea of what is going on.

If no other *User-Agent* is specified, "mastodonpy" will be used.

Mastodon.**log_in**(*username: str | None = None*, *password: str | None = None*, *code: str | None = None*, *redirect_uri: str = 'urn:ietf:wg:oauth:2.0:oob'*, *refresh_token: str | None = None*, *scopes: List[str] = ['read', 'write', 'follow', 'push']*, *to_file: str | PurePath | None = None*, *allow_http: bool = False*) → str

Get the access token for a user, either via OAuth code flow, or (deprecated) password flow.

Will throw a *MastodonIllegalArgumentError* if the OAuth flow data is incorrect, and *MastodonAPIError* if all of the requested scopes were not granted.

For OAuth2, obtain a code via having your user go to the URL returned by *auth_request_url()* and pass it as the code parameter. In this case, make sure to also pass the same redirect_uri parameter as you used when generating the auth request URL, as well as the same set of scopes, or else your auth request will fail. If passing *code`you should not pass `username* or *password*.

When using the password flow, the username is the email address used to log in into Mastodon. **Note that Mastodon has removed this flow starting with 4.4.0, so it is unfortunately not possible to log in in this way anymore. Please use either the code flow, or generate a token from the web UI.**

Can persist access token to file *to_file*, to be used in the constructor. Pass *allow_http* to allow HTTP URLs for the OAuth server, which is recommended only for testing.

Returns the access token as a string.

Mastodon.**auth_request_url**(*client_id: str | PurePath | None = None*, *redirect_uris: str = 'urn:ietf:wg:oauth:2.0:oob'*, *scopes: List[str] = ['read', 'write', 'follow', 'push']*, *force_login: bool = False*, *state: str | None = None*, *lang: str | None = None*, *skip_server_info=False*, *allow_http: bool = False*) → str

Returns the URL that a client needs to request an OAuth grant from the server.

To log in with OAuth, send your user to this URL. The user will then log in and get a code which you can pass to *log_in()*.

scopes are as in *log_in()*, redirect_uris is where the user should be redirected to after authentication. Note that *redirect_uris* must be one of the URLs given during app registration, and that despite the plural-like name, you only get to use one here. When using urn:ietf:wg:oauth:2.0:oob, the code is simply displayed, otherwise it is added to the given URL as the "code" request parameter. Note that if you pass scopes, you MUST pass the same set of scopes to *log_in()* and *create_app() <create_app()>*, otherwise, your auth request will fail.

Pass force_login if you want the user to always log in even when already logged into web Mastodon (i.e. when registering multiple different accounts in an app).

state is the oauth *state* parameter to pass to the server. It is strongly suggested to use a random, nonguessable value (i.e. nothing meaningful and no incrementing ID) to preserve security guarantees. It can be left out for non-web login flows.

Pass an ISO 639-1 (two letter) or, for languages that do not have one, 639-3 (three letter) language code as *lang* to control the display language for the oauth form.

Pass *skip_server_info* to skip retrieving the OAuth authorization server info, in case you want to avoid the extra network request and are confident that the oauth server is at the default location.

Mastodon.**set_language**(*lang: str*)

Set the locale Mastodon will use to generate responses. Valid parameters are all ISO 639-1 (two letter) or, for languages that do not have one, 639-3 (three letter) language codes. This affects some error messages (those related to validation) and trends.

Mastodon.**revoke_access_token**(*allow_http: bool = False*)

Revoke the oauth token the user is currently authenticated with, effectively removing the apps access and requiring the user to log in again.

Mastodon.**create_account**(*username: str*, *password: str*, *email: str*, *agreement: bool = False*, *reason: str | None = None*, *locale: str = 'en'*, *scopes: List[str] = ['read', 'write', 'follow', 'push']*, *to_file: str | None = None*, *return_detailed_error: bool = False*, *date_of_birth: datetime | None = None*) → str | None | Tuple[str | None, *AccountCreationError*]

Creates a new user account with the given username, password and email. "agreement" must be set to true (after showing the user the instance's user agreement and having them agree to it), "locale" specifies the language for the confirmation email as an ISO 639-1 (two letter) or, if a language does not have one, 639-3 (three letter) language code. *reason* can be used to specify why a user would like to join if approved-registrations mode is on. date_of_birth can be used to specify the date of birth of the user, which is required if the server has a minimum age requirement set.

Does not require an access token, but does require a client grant.

By default, this method is rate-limited by IP to 5 requests per 30 minutes.

Returns an access token (just like log_in), which it can also persist to to_file, and sets it internally so that the user is now logged in. Note that this token can only be used after the user has confirmed their email.

By default, the function will throw if the account could not be created. Alternately, when *return_detailed_error* is passed, Mastodon.py will return the detailed error response that the API provides (Starting from version 3.4.0 - not checked here) as an dict with error details as the second return value and the token returned as *None* in case of error. The dict will contain a text *error* values as well as a *details* value which is a dict with one optional key for each potential field (*username*, *password*, *email* and *agreement*), each if present containing a dict with an *error* category and free text *description*. Valid error categories are:

- ERR_BLOCKED - When e-mail provider is not allowed
- ERR_UNREACHABLE - When e-mail address does not resolve to any IP via DNS (MX, A, AAAA)
- ERR_TAKEN - When username or e-mail are already taken
- ERR_RESERVED - When a username is reserved, e.g. "webmaster" or "admin"
- ERR_ACCEPTED - When agreement has not been accepted
- ERR_BLANK - When a required attribute is blank
- ERR_INVALID - When an attribute is malformed, e.g. wrong characters or invalid e-mail address
- ERR_TOO_LONG - When an attribute is over the character limit
- ERR_TOO_SHORT - When an attribute is under the character requirement
- ERR_INCLUSION - When an attribute is not one of the allowed values, e.g. unsupported locale

Added: Mastodon v2.7.0, last changed: Mastodon v2.7.0

`Mastodon.email_resend_confirmation()`

Requests a re-send of the users confirmation mail for an unconfirmed logged in user.

Only available to the app that the user originally signed up with.

Added: Mastodon v3.4.0, last changed: Mastodon v3.4.0

4.7.3 OAuth information

`Mastodon.oauth_authorization_server_info()` → *OAuthServerInfo* | *AttribAccessDict*

Returns the OAuth authorization server information, including the supported grant types. This is useful to determine which authentication methods are available on the server, supported scopes, URLs to make various OAuth requests, to, etc. Mastodon only supports this after version 4.3.0, and alternative implementations may or may not support it, so if aiming for maximum compatibility, you should likely assume it is not present.

Returns an empty dictionary if unsupported by the server.

Technically added in 4.3.0 but we never do a version check to avoid potential complications.

Mastodon.**oauth_userinfo**() → *OAuthUserInfo*

Returns information about the authenticated user.

Intended for something called "OpenID Connect", which you can find information about here: https://openid.net/developers/how-connect-works/

Added: Mastodon v4.3.0, last changed: Mastodon v4.3.0 (parameters), Mastodon v4.4.0 (return value)

4.7.4 User preferences

Mastodon.**preferences**() → *Preferences*

Fetch the user's preferences, which can be used to set some default options. As of 2.8.0, apps can only fetch, not update preferences.

Added: Mastodon v2.8.0, last changed: Mastodon v2.8.0 (parameters), Mastodon v2.8.0 (return value)

4.8 Statuses, media and polls

4.8.1 Statuses

These functions allow you to get information about single statuses and to post and update them, as well as to favourite, bookmark, mute reblog ("boost") and to undo all of those. For status pinning, check out TODO and TODO on the accounts page.

Reading

Mastodon.**status**(*id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Status*

Fetch information about a single toot.

Does not require authentication for publicly visible statuses.

Added: Mastodon v1.0.0, last changed: Mastodon v2.0.0 (parameters), Mastodon v4.4.0 (return value)

Mastodon.**status_context**(*id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Context*

Fetch information about ancestors and descendants of a toot.

Does not require authentication for publicly visible statuses.

Added: Mastodon v1.0.0, last changed: Mastodon v1.0.0 (parameters), Mastodon v0.6.0 (return value)

Mastodon.**status_reblogged_by**(*id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *NonPaginatableList*[*Account*]

Fetch a list of users that have reblogged a status.

Does not require authentication for publicly visible statuses.

Interesting caveat: If you self-reblog a status with private visibility, this endpoint will not return your account as having reblogged it.

Added: Mastodon v1.0.0, last changed: Mastodon v2.1.0

Mastodon.**status_favourited_by**(*id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *NonPaginatableList*[*Account*]

Fetch a list of users that have favourited a status.

Does not require authentication for publicly visible statuses.

Added: Mastodon v1.0.0, last changed: Mastodon v2.1.0

Mastodon.**status_card**(*id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *PreviewCard*

Fetch a card associated with a status. A card describes an object (such as an external video or link) embedded into a status.

Does not require authentication for publicly visible statuses.

This function is deprecated as of 3.0.0 and the endpoint does not exist anymore - you should just use the "card" field of the status instead. Mastodon.py will try to mimic the old behaviour, but this is somewhat inefficient and not guaranteed to be the case forever.

Added: Mastodon v1.0.0, last changed: Mastodon v3.0.0 (parameters), Mastodon v4.3.0 (return value)

Mastodon.**status_history**(*id:* StatusEdit | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *NonPaginatableList*[*StatusEdit*]

Returns the edit history of a status as a list of StatusEdit objects, starting from the original form. Note that this means that a status that has been edited once will have *two* entries in this list, a status that has been edited twice will have three, and so on.

Added: Mastodon v3.5.0, last changed: Mastodon v3.5.0

Mastodon.**status_source**(*id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *StatusSource*

Returns the source of a status for editing.

Return value is a dictionary containing exactly the parameters you could pass to *status_update()* to change nothing about the status, except *status* is *text* instead.

Mastodon.**statuses**(*ids: List[*Status | *str* | *int* | MaybeSnowflakeIdType | *datetime]*) → List[*Status*]

Fetch information from multiple statuses by a list of status *id*.

Does not require authentication for publicly visible accounts.

Added: Mastodon v4.3.0, last changed: Mastodon v4.3.0

Mastodon.**favourites**(*max_id: str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *min_id: str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *since_id: str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *limit: int* | *None* = *None*) → *PaginatableList*[*Status*]

Fetch the logged-in user's favourited statuses.

This endpoint uses internal ids for pagination, passing status ids to *max_id*, *min_id*, or *since_id* will not work.

Added: Mastodon v1.0.0, last changed: Mastodon v2.6.0

Mastodon.**bookmarks**(*max_id: str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *min_id: str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *since_id: str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *limit: int* | *None* = *None*) → *PaginatableList*[*Status*]

Get a list of statuses bookmarked by the logged-in user.

This endpoint uses internal ids for pagination, passing status ids to *max_id*, *min_id*, or *since_id* will not work.

Added: Mastodon v3.1.0, last changed: Mastodon v3.1.0

Writing

Mastodon.**status_post**(*status: str*, *in_reply_to_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *media_ids: List[*MediaAttachment | *str* | *int* | MaybeSnowflakeIdType | *datetime]* | *None* = *None*, *sensitive: bool* = *False*, *visibility: str* | *None* = *None*, *spoiler_text: str* | *None* = *None*, *language: str* | *None* = *None*, *idempotency_key: str* | *None* = *None*, *content_type: str* | *None* = *None*, *scheduled_at: datetime* | *None* = *None*, *poll:* Poll | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *quote_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *strict_content_type: bool* = *False*) → *Status* | *ScheduledStatus*

Post a status. Can optionally be in reply to another status and contain media.

media_ids should be a list. (If it's not, the function will turn it into one.) It can contain up to four pieces of media (uploaded via *media_post()*). *media_ids* can also be the objects returned by *media_post()* - they are unpacked automatically.

The *sensitive* boolean decides whether or not media attached to the post should be marked as sensitive, which hides it by default on the Mastodon web front-end.

The *visibility* parameter is a string value and accepts any of:

- `'direct'` - post will be visible only to **mentioned users**, known in Mastodon's UI as "Mentioned users only"
- `'private'` - post will be visible only to **followers**, known in Mastodon's UI as "Followers only"
- `'unlisted'` - post will be public but **will not appear** on the public timelines
- `'public'` - post will be public and **will appear** on public timelines

If not passed in, *visibility* defaults to match the current account's default-privacy setting (starting with Mastodon version 1.6) or its locked setting - `'private'` if the account is locked, `'public'` otherwise (for Mastodon versions lower than 1.6).

The *spoiler_text* parameter is a string to be shown as a warning before the text of the status. If no text is passed in, no warning will be displayed.

Specify *language* to override automatic language detection. The parameter accepts all valid ISO 639-1 (2-letter) or for languages where that do not have one, 639-3 (three letter) language codes.

You can set *idempotency_key* to a value to uniquely identify an attempt at posting a status. Even if you call this function more than once, if you call it with the same *idempotency_key*, only one status will be created.

Pass a datetime as *scheduled_at* to schedule the toot for a specific time (the time must be at least 5 minutes into the future). If this is passed, status_post returns a *ScheduledStatus* instead.

Pass *poll* to attach a poll to the status. An appropriate object can be constructed using *make_poll()* . Note that as of Mastodon version 2.8.2, you can only have either media or a poll attached, not both at the same time.

You can use *get_status_length()* to count how many characters a status you want to post would take up in terms of Mastodons character limit. The limits can be retrieved from the instance information (*instance_v2()*).

Specific to "pleroma" feature set:: Specify *content_type* to set the content type of your post on Pleroma. It accepts 'text/plain' (default), 'text/markdown', 'text/html' and 'text/bbcode'. This parameter is not supported on Mastodon servers, but will be safely ignored if set. If you want to throw an error if the content type is not known to work on the server, set *strict_content_type* to True.

Specific to "fedibird" feature set:: The *quote_id* parameter is a non-standard extension that specifies the id of a quoted status.

Returns the new status.

Added: Mastodon v1.0.0, last changed: Mastodon v2.8.0

Mastodon.**status_reply**(*to_status: Status | str | int | MaybeSnowflakeIdType | datetime*, *status: str*, *media_ids: List[MediaAttachment | str | int | MaybeSnowflakeIdType | datetime] | None = None*, *sensitive: bool = False*, *visibility: str | None = None*, *spoiler_text: str | None = None*, *language: str | None = None*, *idempotency_key: str | None = None*, *content_type: str | None = None*, *scheduled_at: datetime | None = None*, *poll: Poll | str | int | MaybeSnowflakeIdType | datetime | None = None*, *quote_id: Status | str | int | MaybeSnowflakeIdType | datetime | None = None*, *untag: bool = False*, *strict_content_type: bool = False*) → *Status*

Helper function - acts like status_post, but prepends the name of all the users that are being replied to the status text and retains CW and visibility if not explicitly overridden.

Note that *to_status* must be a *Status* and not just an ID.

Set *untag* to True if you want the reply to only go to the user you are replying to, removing every other mentioned user from the conversation.

Added: Mastodon v1.0.0, last changed: Mastodon v2.8.0 (parameters), Mastodon v4.4.0 (return value)

Mastodon.**toot**(*status: str*) → *Status*

Synonym for *status_post()* that only takes the status text as input.

Usage in production code is not recommended.

Added: Mastodon v1.0.0, last changed: Mastodon v2.8.0 (parameters), Mastodon v4.4.0 (return value)

Mastodon.**make_poll**(*options: List[str]*, *expires_in: int*, *multiple: bool = False*, *hide_totals: bool = False*) → *Poll*

Generate a poll object that can be passed as the *poll* option when posting a status.

options is an array of strings with the poll options (Maximum, by default: 4 - see the instance configuration for the actual value on any given instance, if stated). *expires_in* is the time in seconds for which the poll should be open. Set multiple to True to allow people to choose more than one answer. Set hide_totals to True to hide the results of the poll until it has expired.

Added: Mastodon v2.8.0, last changed: Mastodon v2.8.0 (parameters), Mastodon v2.8.0 (return value)

Mastodon.**status_reblog**(*id: Status | str | int | MaybeSnowflakeIdType | datetime*, *visibility: str | None = None*) → *Status*

Reblog / boost a status.

The visibility parameter functions the same as in *status_post()* and allows you to reduce the visibility of a reblogged status.

Returns a new Status that wraps around the reblogged status.

Added: Mastodon v1.0.0, last changed: Mastodon v2.0.0 (parameters), Mastodon v4.4.0 (return value)

Mastodon.**status_unreblog**(*id: Status | str | int | MaybeSnowflakeIdType | datetime*) → *Status*

Un-reblog a status.

Returns the status that used to be reblogged.

Added: Mastodon v1.0.0, last changed: Mastodon v2.0.0 (parameters), Mastodon v4.4.0 (return value)

Mastodon.**status_favourite**(*id: Status | str | int | MaybeSnowflakeIdType | datetime*) → *Status*

Favourite a status.

Returns the favourited status.

Added: Mastodon v1.0.0, last changed: Mastodon v2.0.0 (parameters), Mastodon v4.4.0 (return value)

Mastodon.**status_unfavourite**(*id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Status*

Un-favourite a status.

Returns the un-favourited status.

Added: Mastodon v1.0.0, last changed: Mastodon v2.0.0 (parameters), Mastodon v4.4.0 (return value)

Mastodon.**status_mute**(*id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Status*

Mute notifications for a status.

Returns the now muted status

Added: Mastodon v1.4.0, last changed: Mastodon v2.0.0 (parameters), Mastodon v4.4.0 (return value)

Mastodon.**status_unmute**(*id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Status*

Unmute notifications for a status.

Returns the status that used to be muted.

Added: Mastodon v1.4.0, last changed: Mastodon v2.0.0 (parameters), Mastodon v4.4.0 (return value)

Mastodon.**status_bookmark**(*id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Status*

Bookmark a status as the logged-in user.

Returns the now bookmarked status

Added: Mastodon v3.1.0, last changed: Mastodon v3.1.0 (parameters), Mastodon v4.4.0 (return value)

Mastodon.**status_unbookmark**(*id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Status*

Unbookmark a bookmarked status for the logged-in user.

Returns the status that used to be bookmarked.

Added: Mastodon v3.1.0, last changed: Mastodon v3.1.0 (parameters), Mastodon v4.4.0 (return value)

Mastodon.**status_delete**(*id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *delete_media: bool = None*) → *Status*

Delete a status

Returns the now-deleted status, with an added "text" attribute that contains the text that was used to compose this status (this can be used to power "delete and redraft" functionality) as well as either poll or media_attachments set in the same way. Note that when reattaching media, you have to wait up to several seconds for the media to be un-attached from the original status - that operation is not synchronous with the delete.

Pass *delete_media=True* to delete the media attachments of the status immediately, instead of just scheduling them for deletion as part of the next media cleanup. If you set this, you will not be able to reuse them in a new status (so if you're delete-redrafting, you should not set this).

Added: Mastodon v1.0.0, last changed: Mastodon v1.0.0 (parameters), Mastodon v4.4.0 (return value)

Mastodon.**status_update**(*id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *status: str*, *spoiler_text: str* | *None = None*, *sensitive: bool* | *None = None*, *media_ids: List[*MediaAttachment | *str* | *int* | MaybeSnowflakeIdType | *datetime]* | *None = None*, *poll:* Poll | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *media_attributes: List[Dict[str, Any]]* | *None = None*) → *Status*

Edit a status. The meanings of the fields are largely the same as in *status_post()*, though not every field can be edited. The *status* parameter is mandatory.

Note that editing a poll will reset the votes.

To edit media metadata, generate a list of dictionaries with the following keys:

Added: Mastodon v3.5.0, last changed: Mastodon v4.1.0 (parameters), Mastodon v4.4.0 (return value)

Mastodon.**generate_media_edit_attributes**(*id:* MediaAttachment | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *description: str* | *None* = *None*, *focus: Tuple[float, float]* | *None* = *None*, *thumbnail: str* | *PurePath* | *IO[bytes]* | *None* = *None*, *thumb_mimetype: str* | *None* = *None*) → Dict[str, Any]

Helper function to generate a single media edit attribute dictionary.

Parameters: - *id* (str): The ID of the media attachment (mandatory). - *description* (Optional[str]): A new description for the media. - *focus* (Optional[Tuple[float, float]]): The focal point of the media. - *thumbnail* (Optional[PathOrFile]): The thumbnail to be used.

4.8.2 Scheduled statuses

These functions allow you to get information about scheduled statuses and to update scheduled statuses that already exist. To create new scheduled statuses, use *status_post()* with the *scheduled_at* parameter.

Reading

Mastodon.**scheduled_statuses**(*max_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *min_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *since_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *limit: int* | *None* = *None*) → *PaginatableList*[*ScheduledStatus*]

Fetch a list of scheduled statuses

Added: Mastodon v2.7.0, last changed: Mastodon v2.7.0

Mastodon.**scheduled_status**(*id:* ScheduledStatus | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *ScheduledStatus*

Fetch information about the scheduled status with the given id.

Added: Mastodon v2.7.0, last changed: Mastodon v2.7.0 (parameters), Mastodon v2.7.0 (return value)

Writing

Mastodon.**scheduled_status_update**(*id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *scheduled_at: datetime*) → *ScheduledStatus*

Update the scheduled time of a scheduled status.

New time must be at least 5 minutes into the future.

Returned object reflects the updates to the scheduled status.

Added: Mastodon v2.7.0, last changed: Mastodon v2.7.0 (parameters), Mastodon v2.7.0 (return value)

Mastodon.**scheduled_status_delete**(*id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime*)

Deletes a scheduled status.

Added: Mastodon v2.7.0, last changed: Mastodon v2.7.0

4.8.3 Media

This function allows you to upload media to Mastodon and update media uploads. The returned media IDs (Up to 4 at the same time on a default configuration Mastodon instance) can then be used with post_status to attach media to statuses.

Mastodon.**media_post**(*media_file: str* | *PurePath* | *IO[bytes]*, *mime_type: str* | *None* = *None*, *description: str* | *None* = *None*, *focus: Tuple[float, float]* | *None* = *None*, *file_name: str* | *None* = *None*, *thumbnail: str* | *PurePath* | *IO[bytes]* | *None* = *None*, *thumbnail_mime_type: str* | *None* = *None*, *synchronous: bool* = *False*) → *MediaAttachment*

Post an image, video or audio file. *media_file* can either be data or a file name. If data is passed directly, the mime type has to be specified manually, otherwise, it is determined from the file name. *focus* should be a tuple of floats between -1 and 1, giving the x and y coordinates of the images focus point for cropping (with the origin being the images center).

Throws a *MastodonIllegalArgumentError* if the mime type of the passed data or file can not be determined properly.

file_name can be specified to upload a file with the given name, which is ignored by Mastodon, but some other Fediverse server software will display it. If no name is specified, a random name will be generated. The filename of a file specified in media_file will be ignored.

Starting with Mastodon 3.2.0, *thumbnail* can be specified in the same way as *media_file* to upload a custom thumbnail image for audio and video files.

When using the v2 API (post Mastodon version 3.1.4), the *url* in the returned dict will be *null*, since attachments are processed asynchronously. You can fetch an updated dict using *media*. Pass "synchronous" to emulate the old behaviour. Not recommended, inefficient and deprecated, will eat your API quota, you know the deal.

The returned value (or its id) can be used in a status post as the *media_ids* parameter.

Added: Mastodon v1.0.0, last changed: Mastodon v3.2.0 (parameters), Mastodon v4.0.0 (return value)

`Mastodon.`**`media_update`**(*id:* MediaAttachment | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *description: str* | *None = None*, *focus: Tuple[float, float]* | *None = None*, *thumbnail: str* | *PurePath* | *IO[bytes]* | *None = None*, *thumbnail_mime_type=None*) → *MediaAttachment*

Update the metadata of the media file with the given *id*. *description* and *focus* and *thumbnail* are as in *media_post()* .

The returned dict reflects the updates to the media attachment.

Note: This is for updating the metadata of an *unattached* media attachment (i.e. one that has not been used in a status yet). For editing media attachment metadata after posting, see *status_update* and *generate_media_edit_attributes*.

Added: Mastodon v2.3.0, last changed: Mastodon v3.2.0 (parameters), Mastodon v4.0.0 (return value)

`Mastodon.`**`media`**(*id:* MediaAttachment | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *MediaAttachment*

Get the updated JSON for one non-attached / in progress media upload belonging to the logged-in user.

Added: Mastodon v3.1.4, last changed: Mastodon v3.1.4 (parameters), Mastodon v4.0.0 (return value)

4.8.4 Polls

This function allows you to get and refresh information about polls as well as to vote in polls

Reading

`Mastodon.`**`poll`**(*id:* Poll | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Poll*

Fetch information about the poll with the given id

Added: Mastodon v2.8.0, last changed: Mastodon v2.8.0 (parameters), Mastodon v2.8.0 (return value)

Writing

`Mastodon.`**`poll_vote`**(*id:* Poll | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *choices: int* | *List[int]*) → *Poll*

Vote in the given poll.

choices is the index of the choice you wish to register a vote for (i.e. its index in the corresponding polls *options* field. In case of a poll that allows selection of more than one option, a list of indices can be passed.

You can only submit choices for any given poll once in case of single-option polls, or only once per option in case of multi-option polls.

The returned object will reflect the updated votes.

Added: Mastodon v2.8.0, last changed: Mastodon v2.8.0 (parameters), Mastodon v2.8.0 (return value)

4.8.5 Translation

These functions allow you to get machine translations for statuses, if the instance supports it.

`Mastodon.status_translate`(*id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *lang: str* | *None* = *None*) → *Translation*

Translate the status content into some language.

Raises a MastodonAPIError if the server can't perform the requested translation, for any reason (doesn't support translation, unsupported language pair, etc.).

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0 (parameters), Mastodon v4.0.0 (return value)

4.9 Accounts, relationships and lists

4.9.1 Accounts

These functions allow you to get information about accounts and associated data as well as update that data - profile data (incuding pinned statuses and endorsements) for the logged in users account, and notes for everyone else

Reading

`Mastodon.account_verify_credentials`() → *Account*

Fetch logged-in user's account information. Returns the version of the Account object with *source* field.

Added: Mastodon v1.0.0, last changed: Mastodon v2.1.0 (parameters), Mastodon v4.2.0 (return value)

`Mastodon.me`() → *Account*

Get this user's account. Synonym for *account_verify_credentials()*, does exactly the same thing, just exists because *account_verify_credentials()* has a confusing name.

Added: Mastodon v1.0.0, last changed: Mastodon v2.1.0 (parameters), Mastodon v4.2.0 (return value)

`Mastodon.account`(*id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Account*

Fetch account information by user *id*.

Does not require authentication for publicly visible accounts.

Added: Mastodon v1.0.0, last changed: Mastodon v1.0.0 (parameters), Mastodon v4.2.0 (return value)

`Mastodon.account_search`(*q: str*, *limit: int* | *None* = *None*, *following: bool* = *False*, *resolve: bool* = *False*, *offset: int* | *None* = *None*) → *NonPaginatableList*[*Account*]

Fetch matching accounts. Will lookup an account remotely if the search term is in the username@domain format and not yet in the database. Set *following* to True to limit the search to users the logged-in user follows.

Paginated in a weird way ("limit" / "offset"), if you want to fetch all results here please do it yourself for now.

Added: Mastodon v1.0.0, last changed: Mastodon v2.8.0

Mastodon.**account_lookup**(*acct: str*) → *Account*

Look up an account from user@instance form (@instance allowed but not required for local accounts). Will only return accounts that the instance already knows about, and not do any webfinger requests. Use *account_search* if you need to resolve users through webfinger from remote.

Added: Mastodon v3.4.0, last changed: Mastodon v3.4.0 (parameters), Mastodon v4.2.0 (return value)

Mastodon.**accounts**(*ids: List[*Account | *str* | *int* | MaybeSnowflakeIdType | *datetime]*) → List[*Account*]

Fetch information from multiple accounts by a list of user *id*.

Does not require authentication for publicly visible accounts.

Added: Mastodon v4.3.0, last changed: Mastodon v4.3.0

Mastodon.**featured_tags**() → *NonPaginatableList*[*FeaturedTag*]

Return the hashtags the logged-in user has set to be featured on their profile.

Added: Mastodon v3.0.0, last changed: Mastodon v3.0.0

Mastodon.**featured_tag_suggestions**() → *NonPaginatableList*[*FeaturedTag*]

Returns the logged-in user's 10 most commonly-used hashtags.

Added: Mastodon v3.0.0, last changed: Mastodon v3.0.0

Mastodon.**account_featured_tags**(*id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *NonPaginatableList*[*Tag*]

Get an account's featured hashtags.

Added: Mastodon v3.3.0, last changed: Mastodon v3.3.0

Mastodon.**endorsements**() → *NonPaginatableList*[*Account*]

Fetch list of users endorsed by the logged-in user.

Added: Mastodon v2.5.0, last changed: Mastodon v2.5.0

Mastodon.**account_statuses**(*id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *only_media: bool = False*, *pinned: bool = False*, *exclude_replies: bool = False*, *exclude_reblogs: bool = False*, *tagged: str | None = None*, *max_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime | None = None*, *min_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime | None = None*, *since_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime | None = None*, *limit: int | None = None*) → *PaginatableList*[*Status*]

Fetch statuses by user *id*. Same options as *timeline()* are permitted. Returned toots are from the perspective of the logged-in user, i.e. all statuses visible to the logged-in user (including DMs) are included.

If *only_media* is set, return only statuses with media attachments. If *pinned* is set, return only statuses that have been pinned. Note that as of Mastodon 2.1.0, this only works properly for instance-local users. If *exclude_replies* is set, filter out all statuses that are replies. If *exclude_reblogs* is set, filter out all statuses that are reblogs. If *tagged* is set, return only statuses that are tagged with *tagged*. Only a single tag without a '#' is valid.

Does not require authentication for Mastodon versions after 2.7.0 (returns publicly visible statuses in that case), for publicly visible accounts.

Added: Mastodon v1.0.0, last changed: Mastodon v2.8.0

Mastodon.**account_familiar_followers**(*id: List[*Account | *str* | *int* | MaybeSnowflakeIdType | *datetime]* | Account | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *NonPaginatableList*[*FamiliarFollowers*]

Find followers for the account given by id (can be a list) that also follow the logged in account.

Added: Mastodon v3.5.0, last changed: Mastodon v3.5.0

Mastodon.**account_lists**(*id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *NonPaginatableList*[*UserList*]

Get all of the logged-in user's lists which the specified user is a member of.

Added: Mastodon v2.1.0, last changed: Mastodon v2.1.0

Writing

Mastodon.**account_update_credentials**(*display_name: str | None = None*, *note: str | None = None*, *avatar: str | PurePath | IO[bytes] | None = None*, *avatar_mime_type: str | None = None*, *header: str | PurePath | IO[bytes] | None = None*, *header_mime_type: str | None = None*, *locked: bool | None = None*, *bot: bool | None = None*, *discoverable: bool | None = None*, *fields: List[Tuple[str, str]] | None = None*, *attribution_domains: List[str] | None = None*) → *Account*

Update the profile for the currently logged-in user.

note is the user's bio.

avatar and 'header' are images. As with media uploads, it is possible to either pass image data and a mime type, or a filename of an image file, for either.

locked specifies whether the user needs to manually approve follow requests.

bot specifies whether the user should be set to a bot.

discoverable specifies whether the user should appear in the user directory.

fields can be a list of up to four name-value pairs (specified as tuples) to appear as semi-structured information in the user's profile.

attribution_domains can be a list of domains that the user wants to allow to attribute content to them.

The returned object reflects the updated account.

Added: Mastodon v1.1.1, last changed: Mastodon v3.1.0 (parameters), Mastodon v4.2.0 (return value)

Mastodon.**account_endorse**(*id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Relationship*

Endorse a user.

The returned object reflects the updated relationship with the user.

Added: Mastodon v4.4.0, last changed: Mastodon v4.4.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**account_unendorse**(*id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Relationship*

Unendorse a user.

The returned object reflects the updated relationship with the user.

Added: Mastodon v4.4.0, last changed: Mastodon v4.4.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**account_note_set**(*id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *comment: str*) → *Relationship*

Set a note (visible to the logged in user only) for the given account.

The returned object contains the updated note.

nb: To retrieve the current note for an account, use *account_relationships*.

Added: Mastodon v3.2.0, last changed: Mastodon v3.2.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**tag_feature**(*name: str*) → *Tag*

Creates a new featured hashtag displayed on the logged-in user's profile.

Same effect as above, but newer. Likely obsoletes *featured_tag_create*.

Added: Mastodon v4.4.0, last changed: Mastodon v4.4.0 (parameters), Mastodon v4.4.0 (return value)

Mastodon.**tag_unfeature**(*name: str*) → *Tag*

Deletes one of the logged-in user's featured hashtags.

Same effect as above, but newer. Likely obsoletes *featured_tag_delete*.

Added: Mastodon v4.4.0, last changed: Mastodon v4.4.0 (parameters), Mastodon v4.4.0 (return value)

Mastodon.**status_pin**(*id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Status*

Pin a status for the logged-in user.

Returns the now pinned status

Added: Mastodon v2.1.0, last changed: Mastodon v2.1.0 (parameters), Mastodon v4.4.0 (return value)

Mastodon.**status_unpin**(*id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Status*

Unpin a pinned status for the logged-in user.

Returns the status that used to be pinned.

Added: Mastodon v2.1.0, last changed: Mastodon v2.1.0 (parameters), Mastodon v4.4.0 (return value)

Mastodon.**account_delete_avatar**()

Delete the logged-in user's avatar.

Added: Mastodon v4.2.0, last changed: Mastodon v4.2.0

Mastodon.**account_delete_header**()

Delete the logged-in user's header.

Added: Mastodon v4.2.0, last changed: Mastodon v4.2.0

Deprecated

Mastodon.**account_pin**(*id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Relationship*

Pin / endorse a user.

The returned object reflects the updated relationship with the user.

Deprecated, use *account_endorse* instead.

Added: Mastodon v2.5.0, last changed: Mastodon v2.5.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**account_unpin**(*id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Relationship*

Unpin / un-endorse a user.

The returned object reflects the updated relationship with the user.

Deprecated, use *account_unendorse* instead.

Added: Mastodon v2.5.0, last changed: Mastodon v2.5.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**featured_tag_create**(*name: str*) → *FeaturedTag*

Creates a new featured hashtag displayed on the logged-in user's profile.

The returned object is the newly featured tag.

Obsoleted by *tag_feature* / *tag_unfeature*.

Added: Mastodon v3.0.0, last changed: Mastodon v3.0.0 (parameters), Mastodon v3.3.0 (return value)

Mastodon.**featured_tag_delete**(*id:* FeaturedTag | *str* | *int* | MaybeSnowflakeIdType | *datetime*)

Deletes one of the logged-in user's featured hashtags.

Obsoleted by *tag_feature* / *tag_unfeature*.

Added: Mastodon v3.0.0, last changed: Mastodon v3.0.0

4.9.2 Following and followers

These functions allow you to get information about the logged in users followers and users that the logged in users follows as well as follow requests and follow suggestions, and to manage that data - most importantly, follow and unfollow users.

Reading

Mastodon.**account_followers**(*id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *max_id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *min_id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *since_id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *limit: int* | *None* = *None*) → *PaginatableList*[*Account*]

Fetch users the given user is followed by.

Added: Mastodon v1.0.0, last changed: Mastodon v2.6.0

Mastodon.**account_following**(*id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *max_id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *min_id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *since_id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *limit: int* | *None* = *None*) → *PaginatableList*[*Account*]

Fetch users the given user is following.

Added: Mastodon v1.0.0, last changed: Mastodon v2.6.0

Mastodon.**account_relationships**(*id: List[*Account | *str* | *int* | MaybeSnowflakeIdType | *datetime]* | Account | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *with_suspended: bool* | *None* = *None*) → *NonPaginatableList*[*Relationship*]

Fetch relationship (following, followed_by, blocking, follow requested) of the logged in user to a given account. *id* can be a list.

Pass *with_suspended* = *True* to include relationships with suspended accounts.

Added: Mastodon v1.0.0, last changed: Mastodon v1.4.0

Mastodon.**follows**(*uri: str*) → *Relationship*

Follow a remote user with username given in username@domain form.

Deprecated - avoid using this. Currently uses a backwards compat implementation that may or may not work properly.

Added: Mastodon v1.0.0, last changed: Mastodon v2.1.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**follow_requests**(*max_id: str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *min_id: str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *since_id: str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *limit: int* | *None* = *None*) → *PaginatableList*[*Account*]

Fetch the logged-in user's incoming follow requests.

Added: Mastodon v1.0.0, last changed: Mastodon v2.6.0

Mastodon.**suggestions**() → *NonPaginatableList*[*Suggestion*] | *NonPaginatableList*[*Account*]

Fetch follow suggestions for the logged-in user.

Will use the v1 endpoint if the server is below 3.4.0, otherwise will use the v2 endpoint and unpack the account dicts.

Writing

Mastodon.**account_follow**(*id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *reblogs: bool = True*, *notify: bool = False*) → *Relationship*

Follow a user.

Set *reblogs* to False to hide boosts by the followed user. Set *notify* to True to get a notification every time the followed user posts.

The returned object reflects the updated relationship with the user.

Added: Mastodon v1.0.0, last changed: Mastodon v3.3.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**account_unfollow**(*id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Relationship*

Unfollow a user.

The returned object reflects the updated relationship with the user.

Added: Mastodon v1.0.0, last changed: Mastodon v1.4.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**follow_request_authorize**(*id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Relationship*

Accept an incoming follow request from the given Account and returns the updated Relationship.

Added: Mastodon v1.0.0, last changed: Mastodon v3.0.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**follow_request_reject**(*id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Relationship*

Reject an incoming follow request from the given Account and returns the updated Relationship.

Added: Mastodon v1.0.0, last changed: Mastodon v3.0.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**suggestion_delete**(*account_id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime*)

Remove the user with the given *account_id* from the follow suggestions.

Added: Mastodon v2.4.3, last changed: Mastodon v2.4.3

4.9.3 Mutes and blocks

These functions allow you to get information about accounts and domains that are muted or blocked by the logged in user, and to block and mute users and domains

Reading

Mastodon.**mutes**(*max_id: str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *min_id: str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *since_id: str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *limit: int* | *None = None*) → *PaginatableList*[*Account*]

Fetch a list of users muted by the logged-in user.

Added: Mastodon v1.1.0, last changed: Mastodon v2.6.0

Mastodon.**blocks**(*max_id: str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *min_id: str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *since_id: str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *limit: int* | *None = None*) → *PaginatableList*[*Account*]

Fetch a list of users blocked by the logged-in user.

Added: Mastodon v1.0.0, last changed: Mastodon v2.6.0

Mastodon.**domain_blocks**(*max_id: str | int | MaybeSnowflakeIdType | datetime | None = None*, *min_id: str | int | MaybeSnowflakeIdType | datetime | None = None*, *since_id: str | int | MaybeSnowflakeIdType | datetime | None = None*, *limit: int | None = None*) → *PaginatableList*[str]

Fetch the logged-in user's blocked domains.

Returns a list of blocked domain URLs (as strings, without protocol specifier).

Added: Mastodon v1.4.0, last changed: Mastodon v2.6.0

Writing

Mastodon.**account_mute**(*id: Account | str | int | MaybeSnowflakeIdType | datetime*, *notifications: bool = True*, *duration: int | None = None*) → *Relationship*

Mute a user.

Set *notifications* to False to receive notifications even though the user is muted from timelines. Pass a *duration* in seconds to have Mastodon automatically lift the mute after that many seconds.

The returned object reflects the updated relationship with the user.

Added: Mastodon v1.1.0, last changed: Mastodon v2.4.3 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**account_unmute**(*id: Account | str | int | MaybeSnowflakeIdType | datetime*) → *Relationship*

Unmute a user.

The returned object reflects the updated relationship with the user.

Added: Mastodon v1.1.0, last changed: Mastodon v1.4.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**account_block**(*id: Account | str | int | MaybeSnowflakeIdType | datetime*) → *Relationship*

Block a user.

The returned object reflects the updated relationship with the user.

Added: Mastodon v1.0.0, last changed: Mastodon v1.4.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**account_unblock**(*id: Account | str | int | MaybeSnowflakeIdType | datetime*) → *Relationship*

Unblock a user.

The returned object reflects the updated relationship with the user.

Added: Mastodon v1.0.0, last changed: Mastodon v1.4.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**account_remove_from_followers**(*id: Account | str | int | MaybeSnowflakeIdType | datetime*) → *Relationship*

Remove a user from the logged in users followers (i.e. make them unfollow the logged in user / "softblock" them).

The returned object reflects the updated relationship with the user.

Added: Mastodon v3.5.0, last changed: Mastodon v3.5.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**domain_block**(*domain: str*)

Add a block for all statuses originating from the specified domain for the logged-in user.

Added: Mastodon v1.4.0, last changed: Mastodon v1.4.0

Mastodon.**domain_unblock**(*domain: str*)

Remove a domain block for the logged-in user.

Added: Mastodon v1.4.0, last changed: Mastodon v1.4.0

4.9.4 Lists

These functions allow you to view information about lists as well as to create and update them. By default, the maximum number of lists for a user is 50.

Reading

Mastodon.**lists**() → *NonPaginatableList*[*UserList*]

Fetch a list of all the Lists by the logged-in user.

Added: Mastodon v2.1.0, last changed: Mastodon v2.1.0

Mastodon.**list**(*id:* UserList | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *UserList*

Fetch info about a specific list.

Added: Mastodon v2.1.0, last changed: Mastodon v2.1.0 (parameters), Mastodon v4.2.0 (return value)

Mastodon.**list_accounts**(*id:* UserList | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *max_id:* UserList | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *min_id:* UserList | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *since_id:* UserList | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *limit: int* | *None* = *None*) → *PaginatableList*[*Account*]

Get the accounts that are on the given list.

Added: Mastodon v2.1.0, last changed: Mastodon v2.6.0

Writing

Mastodon.**list_create**(*title: str*, *replies_policy: str* = *'list'*, *exclusive: bool* = *False*) → *UserList*

Create a new list with the given *title*.

replies_policy controls which replies are shown in the list. It can be one of "followed", "list" or "none". *followed* means that only replies from accounts you follow will be shown, *list* means that only replies from accounts on the list will be shown, and *none* means that no replies will be shown.

Set *exclusive* to True if you want the list to be an *exclusive* list, meaning that accounts on the list will be excluded from the home timeline, appearing exclusively in the list timeline.

Added: Mastodon v2.1.0, last changed: Mastodon v4.2.0 (parameters), Mastodon v4.2.0 (return value)

Mastodon.**list_update**(*id:* UserList | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *title: str*, *replies_policy: str* = *'list'*, *exclusive: bool* = *False*) → *UserList*

Update info about a list, where "info" is really the lists *title*.

replies_policy controls which replies are shown in the list. It can be one of "followed", "list" or "none". *followed* means that only replies from accounts you follow will be shown, *list* means that only replies from accounts on the list will be shown, and *none* means that no replies will be shown.

Set *exclusive* to True if you want the list to be an *exclusive* list, meaning that accounts on the list will be excluded from the home timeline, appearing exclusively in the list timeline.

The returned object reflects the updated list.

Added: Mastodon v2.1.0, last changed: Mastodon v4.2.0 (parameters), Mastodon v4.2.0 (return value)

Mastodon.**list_delete**(*id:* UserList | *str* | *int* | MaybeSnowflakeIdType | *datetime*)

Delete a list.

Added: Mastodon v2.1.0, last changed: Mastodon v2.1.0

Mastodon.**list_accounts_add**(*id:* UserList | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *account_ids:* *List[*Account | *str* | *int* | MaybeSnowflakeIdType | *datetime]*)

Add the account(s) given in *account_ids* to the list.

Added: Mastodon v2.1.0, last changed: Mastodon v2.1.0

Mastodon.**list_accounts_delete**(*id:* UserList | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *account_ids:* *List[*Account | *str* | *int* | MaybeSnowflakeIdType | *datetime]*)

Remove the account(s) given in *account_ids* from the list.

Added: Mastodon v2.1.0, last changed: Mastodon v2.1.0

4.9.5 Following tags

These functions allow you to get information about tags that the logged in user is following and to follow and unfollow tags.

Reading

Mastodon.**followed_tags**(*max_id:* Tag | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *min_id:* Tag | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *since_id:* Tag | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *limit: int* | *None* = *None*) → *PaginatableList*[*Tag*]

Returns the logged-in user's followed tags.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0

Writing

Mastodon.**tag_follow**(*hashtag:* Tag | *str*) → *Tag*

Follow a tag.

Returns the newly followed tag.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0 (parameters), Mastodon v4.4.0 (return value)

Mastodon.**tag_unfollow**(*hashtag:* Tag | *str*) → *Tag*

Unfollow a tag.

Returns the previously followed tag.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0 (parameters), Mastodon v4.4.0 (return value)

4.10 Reading data: Timelines

These functions allow you to access the timelines a logged in user could see, as well as hashtag timelines and the public (federated) and local timelines. For the public, local and hashtag timelines, access is allowed even when not authenticated if the instance admin has enabled this functionality.

Mastodon.**timeline**(*timeline: str* = *'home'*, *max_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *min_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *since_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *limit: int* | *None* = *None*, *only_media: bool* = *False*, *local: bool* = *False*, *remote: bool* = *False*) → *PaginatableList*[*Status*]

Fetch statuses, most recent ones first. *timeline* can be 'home', 'local', 'public', 'tag/<hashtag<', 'list/<id>' or 'link/<url>'. See the following functions documentation for what those do.

The default timeline is the "home" timeline.

Specify *only_media* to only get posts with attached media. Specify *local* to only get local statuses, and *remote* to only get remote statuses. Some options are mutually incompatible as dictated by logic.

May or may not require authentication depending on server settings and what is specifically requested.

See <https://docs.joinmastodon.org/methods/timelines/> for a description of the parameters.

Added: Mastodon v1.0.0, last changed: Mastodon v3.1.4

Mastodon.**timeline_home**(*max_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *min_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *since_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *limit: int* | *None* = *None*, *only_media: bool* = *False*, *local: bool* = *False*, *remote: bool* = *False*) → *PaginatableList*[*Status*]

Convenience method: Fetches the logged-in user's home timeline (i.e. followed users and self). Params as in *timeline()*.

Added: Mastodon v1.0.0, last changed: Mastodon v3.1.4

Mastodon.**timeline_local**(*max_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *min_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *since_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *limit: int* | *None* = *None*, *only_media: bool* = *False*) → *PaginatableList*[*Status*]

Convenience method: Fetches the local / instance-wide timeline, not including replies. Params as in *timeline()*.

Added: Mastodon v1.0.0, last changed: Mastodon v3.1.4

Mastodon.**timeline_public**(*max_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *min_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *since_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *limit: int* | *None* = *None*, *only_media: bool* = *False*, *local: bool* = *False*, *remote: bool* = *False*) → *PaginatableList*[*Status*]

Convenience method: Fetches the public / visible-network / federated timeline, not including replies. Params as in *timeline()*.

Added: Mastodon v1.0.0, last changed: Mastodon v3.1.4

Mastodon.**timeline_hashtag**(*hashtag: str*, *local: bool* = *False*, *max_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *min_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *since_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *limit: int* | *None* = *None*, *only_media: bool* = *False*, *remote: bool* = *False*) → *PaginatableList*[*Status*]

Convenience method: Fetch a timeline of toots with a given hashtag. The hashtag parameter should not contain the leading #. Params as in *timeline()*.

Added: Mastodon v1.0.0, last changed: Mastodon v3.1.4

Mastodon.**timeline_list**(*id:* UserList | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *max_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *min_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *since_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *limit: int* | *None* = *None*, *only_media: bool* = *False*, *local: bool* = *False*, *remote: bool* = *False*) → *PaginatableList*[*Status*]

Convenience method: Fetches a timeline containing all the toots by users in a given list. Params as in *timeline()*.

Added: Mastodon v2.1.0, last changed: Mastodon v3.1.4

Mastodon.**conversations**(*max_id:* Conversation | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *min_id:* Conversation | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *since_id:* Conversation | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *limit: int* | *None* = *None*) → *PaginatableList*[*Conversation*]

Fetches a user's conversations.

Added: Mastodon v2.6.0, last changed: Mastodon v2.6.0

4.11 Instance-wide data and search

4.11.1 Instance information

These functions allow you to fetch information associated with the current instance as well as data from the instance-wide profile directory.

Mastodon.**instance**() → *InstanceV2* | *Instance*

Retrieve basic information about the instance, including the URI and administrative contact email.

Does not require authentication unless locked down by the administrator.

Will return the latest available version of the instance information. If you want a specific one, call the _v1 or _v2 variants

Added: Mastodon v1.1.0, last changed: Mastodon v4.0.0

Mastodon.**instance_v1**() → *Instance*

Retrieve basic information about the instance, including the URI and administrative contact email.

Does not require authentication unless locked down by the administrator.

This is the explicit v1 version of this function. The v2 version is available through instance_v2(). It contains a bit more information than this one, but does not include whether invites are enabled.

Added: Mastodon v1.1.0, last changed: Mastodon v2.3.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**instance_v2**() → *InstanceV2*

Retrieve basic information about the instance, including the URI and administrative contact email.

Does not require authentication unless locked down by the administrator. This is the explicit v2 variant.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0 (parameters), Mastodon v4.3.0 (return value)

Mastodon.**instance_activity**() → *NonPaginatableList*[*Activity*]

Retrieve activity stats about the instance. May be disabled by the instance administrator - throws a MastodonNotFoundError in that case.

Activity is returned for 12 weeks going back from the current week.

Added: Mastodon v2.1.2, last changed: Mastodon v2.1.2

Mastodon.**instance_peers**() → *NonPaginatableList*[str]

Retrieve the instances that this instance knows about. May be disabled by the instance administrator - throws a MastodonNotFoundError in that case.

Returns a list of URL strings.

Added: Mastodon v2.1.2, last changed: Mastodon v2.1.2

Mastodon.**instance_health**() → bool

Basic health check. Returns True if healthy, False if not.

Added: Mastodon v3.0.0, last changed: Mastodon v3.0.0

Mastodon.**instance_nodeinfo**(*schema: str = 'http://nodeinfo.diaspora.software/ns/schema/2.0'*) → *Nodeinfo*

Retrieves the instance's nodeinfo information.

For information on what the nodeinfo can contain, see the nodeinfo specification: https://github.com/jhass/nodeinfo . By default, Mastodon.py will try to retrieve the version 2.0 schema nodeinfo, for which we have a well defined return object. If you go outside of that, all bets are off.

To override the schema, specify the desired schema with the *schema* parameter.

Added: Mastodon v3.0.0, last changed: Mastodon v3.0.0 (parameters), Mastodon v3.0.0 (return value)

Mastodon.**instance_rules**() → *NonPaginatableList*[*Rule*]

Retrieve instance rules.

Added: Mastodon v3.4.0, last changed: Mastodon v3.4.0

Mastodon.**instance_extended_description**() → *ExtendedDescription*

Retrieve the instance's extended description.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**instance_terms_of_service**(*date: date | None = None*) → *TermsOfService*

Retrieve the instance's terms of service.

If *date* is specified, it will return the terms of service that were put in effect on that date.

NB: This is not (currently?) a range lookup, you can only get the terms of service for a specific, exact date.

Added: Mastodon v4.4.0, last changed: Mastodon v4.4.0 (parameters), Mastodon v4.4.0 (return value)

Profile directory

Mastodon.**directory**(*offset: int | None = None*, *limit: int | None = None*, *order: str | None = None*, *local: bool | None = None*) → *NonPaginatableList*[*Account*]

Fetch the contents of the profile directory, if enabled on the server.

offset how many accounts to skip before returning results. Default 0.

limit how many accounts to load. Default 40.

order **"active" to sort by most recently posted statuses (usually the default) or**
"new" to sort by most recently created profiles.

local True to return only local accounts.

Uses offset/limit pagination, not currently handled by the pagination utility functions, do it manually if you have to.

Added: Mastodon v3.0.0, last changed: Mastodon v3.0.0

Emoji

Mastodon.**custom_emojis**() → *NonPaginatableList*[*CustomEmoji*]

Fetch the list of custom emoji the instance has installed.

Does not require authentication unless locked down by the administrator.

Added: Mastodon v2.1.0, last changed: Mastodon v2.1.0

4.11.2 Announcements

These functions allow you to fetch announcements, mark annoucements read and modify reactions.

Reading

Mastodon.**announcements**() → *NonPaginatableList*[*Announcement*]

Fetch currently active announcements.

Added: Mastodon v3.1.0, last changed: Mastodon v3.1.0

Writing

Mastodon.**announcement_dismiss**(*id:* Announcement | *str* | *int* | MaybeSnowflakeIdType | *datetime*)

Set the given annoucement to read.

Added: Mastodon v3.1.0, last changed: Mastodon v3.1.0

Mastodon.**announcement_reaction_create**(*id:* Announcement | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *reaction: str*)

Add a reaction to an announcement. *reaction* can either be a unicode emoji or the name of one of the instances custom emoji.

Will throw an API error if the reaction name is not one of the allowed things or when trying to add a reaction that the user has already added (adding a reaction that a different user added is legal and increments the count).

Added: Mastodon v3.1.0, last changed: Mastodon v3.1.0

Mastodon.**announcement_reaction_delete**(*id:* Announcement | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *reaction: str*)

Remove a reaction to an announcement.

Will throw an API error if the reaction does not exist.

Added: Mastodon v3.1.0, last changed: Mastodon v3.1.0

4.11.3 Trends

These functions, when enabled, allow you to fetch trending tags, statuses and links.

Mastodon.**trending_tags**(*limit: int* | *None* = *None*, *offset: int* | *None* = *None*, *lang: str* | *None* = *None*) → *NonPaginatableList*[*Tag*]

Fetch trending-hashtag information, if the instance provides such information.

Specify *limit* to limit how many results are returned (default 10, the maximum number of results is 20).

Specify *offset* to paginate results. Default 0.

Does not require authentication unless locked down by the administrator.

Important versioning note: This endpoint does not exist for Mastodon versions between 2.8.0 (inclusive) and 3.0.0 (exclusive).

Pass *lang* to override the global locale parameter, which may affect trend ordering.

The results are sorted by the instances's trending algorithm, descending.

Added: Mastodon v3.5.0, last changed: Mastodon v3.5.0

Mastodon.**trending_statuses**(*limit: int | None = None*, *offset: int | None = None*, *lang: str | None = None*) → *NonPaginatableList*[*Status*]

Fetch trending-status information, if the instance provides such information.

Specify *limit* to limit how many results are returned (default 20, the maximum number of results is 40).

Specify *offset* to paginate results. Default 0.

Pass *lang* to override the global locale parameter, which may affect trend ordering.

The results are sorted by the instances's trending algorithm, descending.

Added: Mastodon v3.5.0, last changed: Mastodon v3.5.0

Mastodon.**trending_links**(*limit: int | None = None*, *offset: int | None = None*, *lang: str | None = None*) → *NonPaginatableList*[*PreviewCard*]

Fetch trending-link information, if the instance provides such information.

Specify *limit* to limit how many results are returned (default 10, the maximum number of results is 20).

Specify *offset* to paginate results. Default 0.

The results are sorted by the instances's trending algorithm, descending.

Added: Mastodon v3.5.0, last changed: Mastodon v3.5.0

Mastodon.**trends**(*limit: int | None = None*)

Old alias for *trending_tags()*

Deprecated. Please use *trending_tags()* instead.

Added: Mastodon v2.4.3, last changed: Mastodon v3.5.0

4.11.4 Search

These functions allow you to search for users, tags and, when enabled, full text, by default within your own posts and those you have interacted with.

Mastodon.**search**(*q: str*, *resolve: bool = True*, *result_type: str | None = None*, *account_id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *offset: int | None = None*, *min_id: str | int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *max_id: str | int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *exclude_unreviewed: bool = True*) → *Search* | *SearchV2*

Fetch matching hashtags, accounts and statuses. Will perform webfinger lookups if resolve is True. Full-text search is only enabled if the instance supports it, and is restricted to statuses the logged-in user wrote or was mentioned in.

result_type can be one of "accounts", "hashtags" or "statuses", to only search for that type of object.

Specify *account_id* to only get results from the account with that id.

offset, *min_id* and *max_id* can be used to paginate.

exclude_unreviewed can be used to restrict search results for hashtags to only those that have been reviewed by moderators. It is on by default. When using the v1 search API (pre 2.4.1), it is ignored.

Will use search_v1 (no tags in return values) on Mastodon versions before 2.4.1), search_v2 otherwise. Parameters other than resolve are only available on Mastodon 2.8.0 or above - this function will throw a MastodonVersionError if you try to use them on versions before that. Note that the cached version number will be used for this to avoid uneccesary requests.

Added: Mastodon v1.1.0, last changed: Mastodon v2.8.0

Mastodon.**search_v2**(*q*, *resolve: bool = True*, *result_type: str | None = None*, *account_id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *offset: int | None = None*, *min_id: str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *max_id: str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *exclude_unreviewed: bool = True*) → *SearchV2*

Identical to *search_v1()*, except in that it returns tags as objects, has more parameters, and resolves by default.

For more details documentation, please see *search()*

Added: Mastodon v2.4.1, last changed: Mastodon v2.8.0 (parameters), Mastodon v2.4.1 (return value)

4.11.5 Domain blocks

Mastodon.**instance_domain_blocks**() → *NonPaginatableList*[*DomainBlock*]

Fetch a list of domains that have been blocked by the instance. Public endpoint, requires authentication if limited to users.

Returns a MastodonAPIError if the admin has chosen to not make the list public, or to now show it at all.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0

4.11.6 Translation support

Mastodon.**instance_languages**() → *NonPaginatableList*[*SupportedLocale*]

Fetch a list of languages that the instance supports.

Added: Mastodon v4.2.0, last changed: Mastodon v4.2.0

Mastodon.**instance_translation_languages**() → Dict[str, List[str]]

Retrieve the instance's translation languages.

Returns a dict with language pairs, where the key is the language code and the value is a list of language codes that the instance can translate that language to.

4.12 Notifications and filtering

4.12.1 Notifications

These functions allow you to get information about a user's notifications as well as to clear all or some notifications and to mark conversations as read.

Reading

Mastodon.**notifications**(*id:* Notification | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *account_id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *max_id:* Notification | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *min_id:* Notification | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *since_id:* Notification | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *limit: int | None = None*, *exclude_types: List[str] | None = None*, *types: List[str] | None = None*, *mentions_only: bool | None = None*) → *PaginatableList*[*Notification*] | *Notification*

Fetch notifications (mentions, favourites, reblogs, follows) for the logged-in user. Pass *account_id* to get only notifications originating from the given account.

There are different types of notifications:

- *follow* - A user followed the logged in user

- *follow_request* - A user has requested to follow the logged in user (for locked accounts)
- *favourite* - A user favourited a post by the logged in user
- *reblog* - A user reblogged a post by the logged in user
- *mention* - A user mentioned the logged in user
- *poll* - A poll the logged in user created or voted in has ended
- *update* - A status the logged in user has reblogged (and only those, as of 4.0.0) has been edited
- *status* - A user that the logged in user has enabned notifications for has enabled *notify* (see *account_follow()*)
- *admin.sign_up* - For accounts with appropriate permissions: A new user has signed up
- *admin.report* - For accounts with appropriate permissions: A new report has been received
- *severed_relationships* - Some of the logged in users relationships have been severed due to a moderation action on this server
- *moderation_warning* - The logged in user has been warned by a moderator

Parameters *exclude_types* and *types* are array of these types, specifying them will in- or exclude the types of notifications given. It is legal to give both parameters at the same tine, the result will then be the intersection of the results of both filters. Specifying *mentions_only* is a deprecated way to set *exclude_types* to all but mentions.

Can be passed an *id* to fetch a single notification.

Added: Mastodon v1.0.0, last changed: Mastodon v3.5.0

`Mastodon.`**`notifications_unread_count`**`()` → *UnreadNotificationsCount*

Fetch the number of unread notifications for the logged-in user.

Added: Mastodon v4.3.0, last changed: Mastodon v4.3.0 (parameters), Mastodon v4.3.0 (return value)

Writing

`Mastodon.`**`notifications_clear`**`()`

Clear out a user's notifications

Added: Mastodon v1.0.0, last changed: Mastodon v1.0.0

`Mastodon.`**`notifications_dismiss`**`(`*id:* Notification | *str* | *int* | MaybeSnowflakeIdType | *datetime*`)`

Deletes a single notification

Added: Mastodon v1.3.0, last changed: Mastodon v2.9.2

`Mastodon.`**`conversations_read`**`(`*id:* Conversation | *str* | *int* | MaybeSnowflakeIdType | *datetime*`)`

Marks a single conversation as read.

The returned object reflects the conversation's new read status.

Added: Mastodon v2.6.0, last changed: Mastodon v2.6.0

4.12.2 Grouped notifications

This is the more modern notification API, which delivers notifications grouped.

Mastodon.**grouped_notifications**(*max_id: str | int | MaybeSnowflakeIdType | datetime | None = None*, *since_id: str | int | MaybeSnowflakeIdType | datetime | None = None*, *min_id: str | int | MaybeSnowflakeIdType | datetime | None = None*, *limit: int | None = None*, *types: List[str] | None = None*, *exclude_types: List[str] | None = None*, *account_id: Account | str | int | MaybeSnowflakeIdType | datetime | None = None*, *expand_accounts: str | None = 'partial_avatars'*, *grouped_types: List[str] | None = None*, *include_filtered: bool | None = None*) → *GroupedNotificationsResults*

Fetch grouped notifications for the user. Requires scope *read:notifications*.

For base parameters, see *notifications()*.

grouped_types controls which notication types can be grouped together - all, if not specified. NB: "all" here means favourite, follow and reblog - other types are not groupable and are returned individually (with a unique group key) always.

Pass *include_filtered=True* to include filtered notifications in the response.

Pass *expand_accounts="full"* to include full account details in the response, or "partial_avatars" to include a smaller set of account details (in the *partial_accounts* field) for some (but not all - the most recent account triggering a notification is always returned in full) of the included accounts. The default is partial_avatars.

Added: Mastodon v4.3.0, last changed: Mastodon v4.3.0 (parameters), Mastodon v4.3.0 (return value)

Mastodon.**grouped_notification**(*group_key: str*) → *GroupedNotificationsResults*

Fetch details of a single grouped notification by its group key. Requires scope *read:notifications*.

Added: Mastodon v4.3.0, last changed: Mastodon v4.3.0 (parameters), Mastodon v4.3.0 (return value)

Mastodon.**dismiss_grouped_notification**(*group_key: str*) → None

Dismiss a single grouped notification. Requires scope *write:notifications*.

Added: Mastodon v4.3.0, last changed: Mastodon v4.3.0

Mastodon.**grouped_notification_accounts**(*group_key: str*) → *NonPaginatableList*[*Account*]

Fetch accounts associated with a grouped notification. Requires scope *write:notifications*.

Added: Mastodon v4.3.0, last changed: Mastodon v4.3.0

Mastodon.**unread_grouped_notifications_count**(*limit: int | None = None*, *types: List[str] | None = None*, *exclude_types: List[str] | None = None*, *account_id: Account | str | int | MaybeSnowflakeIdType | datetime | None = None*, *grouped_types: List[str] | None = None*) → int

Fetch the count of unread grouped notifications. Requires scope *read:notifications*.

For parameters, see *notifications()* and *grouped_notifications()*.

Added: Mastodon v4.3.0, last changed: Mastodon v4.3.0

4.12.3 Source filtering for notifications

These functions allow you to get information about source filters as well as to create and update filters, and to accept or reject notification requests for filtered notifications.

Mastodon.**notifications_policy**() → *NotificationPolicy*

Fetch the user's notification filtering policy. Requires scope *read:notifications*.

Added: Mastodon v4.3.0, last changed: Mastodon v4.3.0 (parameters), Mastodon v4.3.0 (return value)

Mastodon.**update_notifications_policy**(*for_not_following: str | None = None*, *for_not_followers: str | None = None*, *for_new_accounts: str | None = None*, *for_private_mentions: str | None = None*, *for_limited_accounts: str | None = None*) → *NotificationPolicy*

Update the user's notification filtering policy. Requires scope *write:notifications*.

- *for_not_following*: "accept", "filter", or "drop" notifications from non-followed accounts.
- *for_not_followers*: "accept", "filter", or "drop" notifications from non-followers.
- *for_new_accounts*: "accept", "filter", or "drop" notifications from accounts created in the past 30 days.
- *for_private_mentions*: "accept", "filter", or "drop" notifications from private mentions.
- *for_limited_accounts*: "accept", "filter", or "drop" notifications from accounts limited by moderators.

Added: Mastodon v4.3.0, last changed: Mastodon v4.3.0 (parameters), Mastodon v4.3.0 (return value)

Mastodon.**notification_requests**(*max_id: str | int | MaybeSnowflakeIdType | datetime | None = None*, *since_id: str | int | MaybeSnowflakeIdType | datetime | None = None*, *min_id: str | int | MaybeSnowflakeIdType | datetime | None = None*, *limit: int | None = None*) → *PaginatableList*[*NotificationRequest*]

Fetch notification requests filtered by the user's policy. Requires scope *read:notifications*.

NB: Notification requests are what happens when the user has set their policy to filter notifications from some source.

Added: Mastodon v4.3.0, last changed: Mastodon v4.3.0

Mastodon.**notification_request**(*id:* NotificationRequest | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *NotificationRequest*

Fetch a single notification request by ID. Requires scope *read:notifications*.

Added: Mastodon v4.3.0, last changed: Mastodon v4.3.0 (parameters), Mastodon v4.3.0 (return value)

Mastodon.**accept_notification_request**(*id:* NotificationRequest | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → None

Accept a notification request. This moves filtered notifications from a user back into the main notifications feed and allows future notifications from them. Requires scope *write:notifications*.

Added: Mastodon v4.3.0, last changed: Mastodon v4.3.0

Mastodon.**dismiss_notification_request**(*id:* NotificationRequest | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → None

Dismiss a notification request, removing it from pending requests. Requires scope *write:notifications*.

Added: Mastodon v4.3.0, last changed: Mastodon v4.3.0

Mastodon.**accept_multiple_notification_requests**(*ids: List[*NotificationRequest | *str* | *int* | MaybeSnowflakeIdType | *datetime]*) → None

Accept multiple notification requests at once. This moves filtered notifications from those users back into the main notifications feed and allows future notifications from them. Requires scope *write:notifications*.

Added: Mastodon v4.3.0, last changed: Mastodon v4.3.0

Mastodon.**dismiss_multiple_notification_requests**(*ids: List[*NotificationRequest | *str* | *int* | MaybeSnowflakeIdType | *datetime]*) → None

Dismiss multiple notification requests, removing them from pending requests. Requires scope *write:notifications*.

Added: Mastodon v4.3.0, last changed: Mastodon v4.3.0

Mastodon.**notifications_merged**() → bool

Check whether accepted notification requests have been merged into the main notification feed. Accepting a notification request schedules a background job that merges the filtered notifications. Clients can poll this endpoint to check if the merge has completed. Requires scope *read:notifications*.

Added: Mastodon v4.3.0, last changed: Mastodon v4.3.0

4.12.4 Keyword Filters (v2)

These functions allow you to get information about keyword filters as well as to create and update filters.

NB: The filters are checked server side, but the server still returns all statuses to the client, just with a *filtered* attribute. Filtered notifications most likely end up as notification requests, but I have not validated this.

Mastodon.**filters_v2**() → *NonPaginatableList*[*FilterV2*]

Fetch all filters for the authenticated user.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0

Mastodon.**filter_v2**(*filter_id:* Filter | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Filter*

Fetch a specific filter by its ID.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0 (parameters), Mastodon v3.1.0 (return value)

Mastodon.**create_filter_v2**(*title: str*, *context: List[str]*, *filter_action: str*, *expires_in: int* | *None* = *None*, *keywords_attributes: List[Dict[str, str* | *bool]]* | *None* = *None*) → *FilterV2*

Create a new filter with the given parameters.

title is a human readable name for the filter.

***context* is list of contexts where the filter should apply. Valid values are:**

- "home": Filter applies to the home timeline.
- "notifications": Filter applies to notifications. Filtered notifications land in notification requests.
- "public": Filter applies to the public timelines.
- "thread": Filter applies to conversations.
- "account": Filter applies to account timelines.

***filter_action* gives the policy to be applied when the filter is matched. Valid values are:**

- "warn": The user is warned if the content matches the filter.
- "hide": The content is completely hidden if it matches the filter.

NB: Even if you specify "hide", the status will still be returned - it will just have the "filtered" attribute set.

pass a number of seconds as *expires_in* to make the filter expire in that many seconds. Use None for no expiration.

pass a list of keyword dicts to initially as *keywords_attributes*, each with the following values:

- "keyword": The term to filter on.
- "whole_word": Whether word boundaries should be considered.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**update_filter_v2**(*filter_id:* FilterV2 | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *title: str* | *None* = *None*, *context: List[str]* | *None* = *None*, *filter_action: str* | *None* = *None*, *expires_in: int* | *None* = *None*, *keywords_attributes: List[Dict[str, str* | *bool* | *int]]* | *None* = *None*) → *FilterV2*

Update an existing filter with the given parameters.

Parameters are as in *create_filter_v2()*. Only the parameters you want to update need to be provided.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**delete_filter_v2**(*filter_id:* FilterV2 | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → None

Delete an existing filter.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0

Mastodon.**filter_keywords_v2**(*filter_id:* FilterV2 | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *NonPaginatableList*[*FilterKeyword*]

Fetch all keywords associated with a given filter.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0

Mastodon.**add_filter_keyword_v2**(*filter_id:* FilterV2 | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *keyword: str*, *whole_word: bool = False*) → *FilterKeyword*

Add a single keyword to an existing filter.

Parameters are as in *create_filter_v2() keywords_attributes*.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**delete_filter_keyword_v2**(*keyword_id:* FilterKeyword | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → None

Delete a single keyword from any filter.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0

Mastodon.**filter_statuses_v2**(*filter_id:* FilterV2 | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → List[*FilterStatus*]

Retrieve all status-based filters for a FilterV2.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0

Mastodon.**add_filter_status_v2**(*filter_id:* FilterV2 | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *status_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *FilterStatus*

Add a status to a filter, which will then match on that status in addition to any keywords. Includes reblogs, does not include replies.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**filter_status_v2**(*filter_status_id:* FilterStatus | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *FilterStatus*

Fetch a single status-based filter by its ID.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**delete_filter_status_v2**(*filter_status_id:* FilterStatus | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → None

Remove a status filter from a FilterV2.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0

4.12.5 Push notifications

Mastodon supports the delivery of notifications via webpush.

These functions allow you to manage webpush subscriptions and to decrypt received pushes. Note that the intended setup is not Mastodon pushing directly to a user's client - the push endpoint should usually be a relay server that then takes care of delivering the (encrypted) push to the end user via some mechanism, where it can then be decrypted and displayed.

Mastodon allows an application to have one webpush subscription per user at a time.

All crypto utilities require Mastodon.py's optional "webpush" feature dependencies (specifically, the "cryptography" and "http_ece" packages).

`Mastodon.`**`push_subscription`**`()` → *WebPushSubscription*

Fetch the current push subscription the logged-in user has for this app.

Only one webpush subscription can be active at a time for any given app.

Added: Mastodon v2.4.0, last changed: Mastodon v2.4.0 (parameters), Mastodon v4.4.0 (return value)

`Mastodon.`**`push_subscription_set`**`(`*endpoint: str*, *encrypt_params: Dict[str, str]*, *follow_events: bool | None = None*, *favourite_events: bool | None = None*, *reblog_events: bool | None = None*, *mention_events: bool | None = None*, *poll_events: bool | None = None*, *follow_request_events: bool | None = None*, *status_events: bool | None = None*, *policy: str = 'all'*, *update_events: bool | None = None*, *admin_sign_up_events: bool | None = None*, *admin_report_events: bool | None = None*, *standard: bool = None*`)` → *WebPushSubscription*

Sets up or modifies the push subscription the logged-in user has for this app.

endpoint is the endpoint URL mastodon should call for pushes. Note that mastodon requires https for this URL. *encrypt_params* is a dict with key parameters that allow the server to encrypt data for you: A public key *pubkey* and a shared secret *auth*. You can generate this as well as the corresponding private key using the *push_subscription_generate_keys()* function.

policy controls what sources will generate webpush events. Valid values are *all*, *none*, *follower* and *followed*.

The rest of the parameters controls what kind of events you wish to subscribe to. Events whose names start with "admin" require admin privileges to subscribe to.

- *follow_events* controls whether you receive events when someone follows the logged in user.
- *favourite_events* controls whether you receive events when someone favourites one of the logged in users statuses.
- *reblog_events* controls whether you receive events when someone boosts one of the logged in users statuses.
- *mention_events* controls whether you receive events when someone mentions the logged in user in a status.
- *poll_events* controls whether you receive events when a poll the logged in user has voted in has ended.
- *follow_request_events* controls whether you receive events when someone requests to follow the logged in user.
- *status_events* controls whether you receive events when someone the logged in user has subscribed to notifications for posts a new status.
- *update_events* controls whether you receive events when a status that the logged in user has boosted has been edited.
- *admin_sign_up_events* controls whether you receive events when a new user signs up.
- *admin_report_events* controls whether you receive events when a new report is received.

Pass *standard=True* to use the standard webpush subscription format, instead of the pre-release RFC format mastodon was using before.

Added: Mastodon v2.4.0, last changed: Mastodon v4..0 (parameters), Mastodon v4.4.0 (return value)

Mastodon.**push_subscription_update**(*follow_events: bool | None = None*, *favourite_events: bool | None = None*, *reblog_events: bool | None = None*, *mention_events: bool | None = None*, *poll_events: bool | None = None*, *follow_request_events: bool | None = None*, *status_events: bool | None = None*, *policy: str | None = 'all'*, *update_events: bool | None = None*, *admin_sign_up_events: bool | None = None*, *admin_report_events: bool | None = None*) → *WebPushSubscription*

Modifies what kind of events the app wishes to subscribe to.

Parameters are as in push_subscription_set().

Returned object reflects the updated push subscription.

Added: Mastodon v2.4.0, last changed: Mastodon v2.4.0 (parameters), Mastodon v4.4.0 (return value)

Mastodon.**push_subscription_generate_keys**() → Tuple[Dict[str, str], Dict[str, str]]

Generates a private key, public key and shared secret for use in webpush subscriptions.

Returns two dicts: One with the private key and shared secret and another with the public key and shared secret.

Mastodon.**push_subscription_decrypt_push**(*data: bytes*, *decrypt_params: Dict[str, str]*, *encryption_header: str*, *crypto_key_header: str*) → *PushNotification*

Decrypts *data* received in a webpush request. Requires the private key dict from *push_subscription_generate_keys()* (*decrypt_params*) as well as the Encryption and server Crypto-Key headers from the received webpush

Added: Mastodon v2.4.0, last changed: Mastodon v2.4.0 (parameters), Mastodon v2.4.0 (return value)

Usage example

This is a minimal usage example for the push API, including a small http server to receive webpush notifications.

```
api = Mastodon(...)
keys = api.push_subscription_generate_keys()
api.push_subscription_set(endpoint, keys[1], mention_events=1)

class Handler(http.server.BaseHTTPRequestHandler):
    def do_POST(self):
        self.send_response(201)
        self.send_header('Location', '')  # Mastodon doesn't seem to care about this
        self.end_headers()
        data = self.rfile.read(int(self.headers['content-length']))
        np = api.push_subscription_decrypt_push(data, keys[0], self.headers['Encryption
↪'], self.headers['Crypto-Key'])
        n = api.notifications(id=np.notification_id)
        s = n.status
        self.log_message('\nFrom: %s\n%s', s.account.acct, s.content)
httpd = http.server.HTTPServer(('', 42069), Handler)

try:
    httpd.serve_forever()
except KeyboardInterrupt:
```

(continues on next page)

(continued from previous page)

```
    pass
finally:
    httpd.server_close()
    api.push_subscription_delete()
```

4.12.6 Keyword filters (v1, deprecated)

These functions allow you to get information about keyword filters as well as to create and update filters.

These APIs are deprecated in favor of the v2 APIs - I would recommend using those instead.

Reading

`Mastodon.filters()` → *NonPaginatableList*[*Filter*]

Fetch all of the logged-in user's filters.

Added: Mastodon v2.4.3, last changed: Mastodon v2.4.3

`Mastodon.filter`(*id:* Filter | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Filter*

Fetches information about the filter with the specified *id*.

Added: Mastodon v2.4.3, last changed: Mastodon v2.4.3 (parameters), Mastodon v3.1.0 (return value)

`Mastodon.filters_apply`(*objects:* PaginatableList*[*Status*]* | PaginatableList*[*Notification*]*, *filters:* NonPaginatableList*[*Filter*]* | NonPaginatableList*[*FilterV2*]*, *context: str*) → *PaginatableList*[*Status*] | *PaginatableList*[*Notification*]

Helper function: Applies a list of filters to a list of either statuses or notifications and returns only those matched by none. This function will apply all filters that match the context provided in *context*, i.e. if you want to apply only notification-relevant filters, specify 'notifications'. Valid contexts are 'home', 'notifications', 'public' and 'thread'.

NB: This is for v1 filters. v2 filters are applied by the server, which adds the "filtered" attribute to filtered statuses.

Added: Mastodon v2.4.3, last changed: Mastodon v2.4.3

Writing

`Mastodon.filter_create`(*phrase: str*, *context: str*, *irreversible: bool = False*, *whole_word: bool = True*, *expires_in: int | None = None*) → *Filter*

Creates a new keyword filter. *phrase* is the phrase that should be filtered out, *context* specifies from where to filter the keywords. Valid contexts are 'home', 'notifications', 'public' and 'thread'.

Set *irreversible* to True if you want the filter to just delete statuses server side. This works only for the 'home' and 'notifications' contexts.

Set *whole_word* to False if you want to allow filter matches to start or end within a word, not only at word boundaries.

Set *expires_in* to specify for how many seconds the filter should be kept around.

Returns the newly created filter.

Added: Mastodon v2.4.3, last changed: Mastodon v2.4.3 (parameters), Mastodon v3.1.0 (return value)

`Mastodon.filter_update`(*id:* Filter | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *phrase: str | None = None*, *context: str | None = None*, *irreversible: bool | None = None*, *whole_word: bool | None = None*, *expires_in: int | None = None*) → *Filter*

Updates the filter with the given *id*. Parameters are the same as in *filter_create()*.

Returns the updated filter.

Added: Mastodon v2.4.3, last changed: Mastodon v2.4.3 (parameters), Mastodon v3.1.0 (return value)

`Mastodon.filter_delete`(*id:* Filter | FilterV2 | *str* | *int* | MaybeSnowflakeIdType | *datetime*)

Deletes the filter with the given *id*.

Added: Mastodon v2.4.3, last changed: Mastodon v2.4.3

4.13 Streaming

These functions allow access to the streaming API. Since Mastodon v4.2.0 the support for anonymous streaming api access was dropped. You need to generate an access token now.

If *run_async* is False, these methods block forever (or until an error is encountered).

If *run_async* is True, the listener will listen on another thread and these methods will return a handle corresponding to the open connection. If, in addition, *reconnect_async* is True, the thread will attempt to reconnect to the streaming API if any errors are encountered, waiting *reconnect_async_wait_sec* seconds between reconnection attempts. Note that no effort is made to "catch up" - events created while the connection is broken will not be received. If you need to make sure to get absolutely all notifications / deletes / toots, you will have to do that manually, e.g. using the *on_abort* handler to fill in events since the last received one and then reconnecting. Both *run_async* and *reconnect_async* default to false, and you'll have to set each to true separately to get the behaviour described above.

The connection may be closed at any time by calling the handles close() method. The current status of the handler thread can be checked with the handles is_alive() function, and the streaming status can be checked by calling is_receiving().

The streaming functions take instances of *StreamListener* as the *listener* parameter. A *CallbackStreamListener* class that allows you to specify function callbacks directly is included for convenience.

For new well-known events implement the streaming function in *StreamListener* or *CallbackStreamListener*. The function name is *on_* + the event name. If the event name contains dots, they are replaced with underscored, e.g. for an event called 'status.update' the listener function should be named *on_status_update*.

It may be that future Mastodon versions will come with completely new (unknown) event names. If you want to do something when such an event is received, override the listener function *on_unknown_event*. This has an additional parameter *name* which informs about the name of the event. *unknown_event* contains the content of the event. Alternatively, a callback function can be passed in the *unknown_event_handler* parameter in the *CallbackStreamListener* constructor.

Note that the *unknown_event* handler is *not* guaranteed to receive events once they have been implemented. Events will only go to this handler temporarily, while Mastodon.py has not been updated. Changes to what events do and do not go into the handler will not be considered a breaking change. If you want to handle a new event whose name you _do_ know, define an appropriate handler in your StreamListener, which will work even if it is not listed here.

When in not-async mode or async mode without async_reconnect, the stream functions may raise various exceptions: *MastodonMalformedEventError* if a received event cannot be parsed and *MastodonNetworkError* if any connection problems occur.

Mastodon.py currently does not support websocket based, multiplexed streams, but might in the future.

4.13.1 Stream endpoints

`Mastodon.stream_user`(*listener*, *run_async=False*, *timeout=300*, *reconnect_async=False*, *reconnect_async_wait_sec=5*)

Streams events that are relevant to the authorized user, i.e. home timeline and notifications.

Added: Mastodon v1.1.0, last changed: Mastodon v1.4.2

Mastodon.**stream_public**(*listener*, *run_async=False*, *timeout=300*, *reconnect_async=False*, *reconnect_async_wait_sec=5*, *local=False*, *remote=False*)

Streams public events.

Set *local* to True to only get local statuses. Set *remote* to True to only get remote statuses.

Added: Mastodon v1.1.0, last changed: Mastodon v1.4.2

Mastodon.**stream_local**(*listener*, *run_async=False*, *timeout=300*, *reconnect_async=False*, *reconnect_async_wait_sec=5*)

Streams local public events.

This function is deprecated. Please use stream_public() with parameter *local* set to True instead.

Added: Mastodon v1.1.0, last changed: Mastodon v1.4.2

Mastodon.**stream_hashtag**(*tag*, *listener*, *local=False*, *run_async=False*, *timeout=300*, *reconnect_async=False*, *reconnect_async_wait_sec=5*)

Stream for all public statuses for the hashtag 'tag' seen by the connected instance.

Set *local* to True to only get local statuses.

Added: Mastodon v1.1.0, last changed: Mastodon v1.4.2

Mastodon.**stream_list**(*id*, *listener*, *run_async=False*, *timeout=300*, *reconnect_async=False*, *reconnect_async_wait_sec=5*)

Stream events for the current user, restricted to accounts on the given list.

Added: Mastodon v2.1.0, last changed: Mastodon v2.1.0

Mastodon.**stream_healthy**() → bool

Returns True if streaming API is okay, False or raises an error otherwise.

Added: Mastodon v2.5.0, last changed: Mastodon v2.5.0

4.13.2 StreamListener

class mastodon.**StreamListener**

Callbacks for the streaming API. Create a subclass, override the on_xxx methods for the kinds of events you're interested in, then pass an instance of your subclass to Mastodon.user_stream(), Mastodon.public_stream(), or Mastodon.hashtag_stream().

StreamListener.**on_update**(*status:* Status)

A new status has appeared. *status* is the parsed *status dict* describing the status.

StreamListener.**on_notification**(*notification:* Notification)

A new notification. *notification* is the object describing the notification. For more information, see the documentation for the notifications() method.

StreamListener.**on_delete**(*status_id: str | int |* MaybeSnowflakeIdType *| datetime*)

A status has been deleted. *status_id* is the status' integer ID.

StreamListener.**on_conversation**(*conversation:* Conversation)

A direct message (in the direct stream) has been received. *conversation* is the parsed *conversation dict* dictionary describing the conversation

StreamListener.**on_status_update**(*status:* Status)

A status has been edited. 'status' is the parsed JSON dictionary describing the updated status.

StreamListener.**on_unknown_event**(*name*, *unknown_event=None*)

An unknown mastodon API event has been received. The name contains the event-name and unknown_event contains the content of the unknown event.

StreamListener.**on_abort**(*err*)

There was a connection error, read timeout or other error fatal to the streaming connection. The exception object about to be raised is passed to this function for reference.

Note that the exception will be raised properly once you return from this function, so if you are using this handler to reconnect, either never return or start a thread and then catch and ignore the exception.

StreamListener.**handle_heartbeat**()

The server has sent us a keep-alive message. This callback may be useful to carry out periodic housekeeping tasks, or just to confirm that the connection is still open.

CallbackStreamListener

class mastodon.**CallbackStreamListener**(*update_handler=None*, *local_update_handler=None*, *delete_handler=None*, *notification_handler=None*, *conversation_handler=None*, *unknown_event_handler=None*, *status_update_handler=None*, *filters_changed_handler=None*, *announcement_handler=None*, *announcement_reaction_handler=None*, *announcement_delete_handler=None*, *encryted_message_handler=None*)

Simple callback stream handler class. Can optionally additionally send local update events to a separate handler. Define an unknown_event_handler for new Mastodon API events. This handler is *not* guaranteed to receive these events forever, and should only be used for diagnostics.

4.14 Misc: Markers, reports

4.14.1 Markers

These functions allow you to interact with the timeline "last read" markers, to allow for persisting where the user was reading a timeline between sessions and clients / devices.

Reading

Mastodon.**markers_get**(*timeline: str | List[str] = ['home']*) → Dict[str, *Marker*]

Get the last-read-location markers for the specified timelines. Valid timelines are *home* (the home timeline) and *notifications* (the notifications timeline, affects which notifications are considered read).

Note that despite the singular name, *timeline* can be a list.

Returns a dict with the markers, keyed by timeline name.

Added: Mastodon v3.0.0, last changed: Mastodon v3.0.0

Writing

Mastodon.**markers_set**(*timelines: str | List[str]*, *last_read_ids:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *List[*Status*]* | *List[str* | *int* | MaybeSnowflakeIdType | *datetime]*) → Dict[str, *Marker*]

Set the "last read" marker(s) for the given timeline(s) to the given id(s)

Valid timelines are *home* (the home timeline) and *notifications* (the notifications timeline, affects which notifications are considered read).

Note that if you give an invalid timeline name, this will silently do nothing.

Returns a dict with the updated markers, keyed by timeline name.

Added: Mastodon v3.0.0, last changed: Mastodon v3.0.0

4.14.2 Reports

Reading

In Mastodon versions before 2.5.0 this function allowed for the retrieval of reports filed by the logged in user. It has since been removed.

`Mastodon.reports()` → *NonPaginatableList*[*Report*]

Fetch a list of reports made by the logged-in user.

Warning: This method has now finally been removed, and will not work on Mastodon versions 2.5.0 and above.

Added: Mastodon v1.1.0, last changed: Mastodon v1.1.0

Writing

This function allows you to report a user to the instance moderators as well as to the users home instance.

`Mastodon.report`(*account_id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *status_ids:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *comment: str* | *None* = *None*, *forward: bool* = *False*, *category: str* | *None* = *None*, *rule_ids: List[*Rule | *str* | *int* | MaybeSnowflakeIdType | *datetime]* | *None* = *None*, *forward_to_domains: List[str]* | *None* = *None*) → *Report*

Report statuses to the instances administrators.

Accepts a list of toot IDs associated with the report, and a comment.

Starting with Mastodon 3.5.0, you can also pass a *category* (one out of "spam", "violation" or "other") and *rule_ids* (a list of rule IDs corresponding to the rules returned by the *instance()* API).

Set *forward* to True to forward a report of a remote user to that users instance as well as sending it to the instance local administrators. Set forward_to_domains to a list of domains to forward the report to (only domains of people mentioned in the status), or omitto forward to the domain of the reported status.

Added: Mastodon v1.1.0, last changed: Mastodon v3.5.0 (parameters), Mastodon v4.0.0 (return value)

4.15 Utility: Pagination, Blurhash, Other Utilities

4.15.1 Pagination

These functions allow for convenient retrieval of paginated data.

`Mastodon.fetch_next`(*previous_page:* PaginatableList*[*Entity*]* | Entity | *PaginationInfo*) → *PaginatableList*[*Entity*] | *Entity* | None

Fetches the next page of results of a paginated request. Pass in the previous page in its entirety, or the pagination information dict returned as a part of that pages last status ('_pagination_next').

Returns the next page or None if no further data is available.

`Mastodon.fetch_previous`(*next_page:* PaginatableList*[*Entity*]* | Entity | *PaginationInfo*) → *PaginatableList*[*Entity*] | *Entity* | None

Fetches the previous page of results of a paginated request. Pass in the previous page in its entirety, or the pagination information dict returned as a part of that pages first status ('_pagination_prev').

Returns the previous page or None if no further data is available.

Mastodon.**fetch_remaining**(*first_page:* PaginatableList*[*Entity*]*) → *PaginatableList*[*Entity*]

Fetches all the remaining pages of a paginated request starting from a first page and returns the entire set of results (including the first page that was passed in) as a big list.

Be careful, as this might generate a lot of requests, depending on what you are fetching, and might cause you to run into rate limits very quickly.

Does not work with grouped notifications, since they use a somewhat weird, inside-out pagination scheme. If you need to access these in a paginated way, use fetch_next and fetch_previous directly.

Mastodon.**pagination_iterator**(*start_page:* PaginatableList*[*Entity*]* | *PaginationInfo*, *direction: str = 'next'*, *return_pagination_info: bool = False*) → Iterator[*Entity*]

Returns an iterator that will yield all entries in a paginated request, starting from the given start_page (can also be just the PaginationInfo, in which case the first returned thing will be the result of fetch_next or fetch_previous, depending on the direction). and fetching new pages as needed, and breaks when no more pages are available.

Set direction to "next" to iterate forward, or "previous" to iterate backwards.

If return_pagination_info is True, the iterator will instead yield tuples of (Entity, PaginationInfo), where PaginationInfo is a dictionary containing pagination information for the current page and direction.

Does not work with grouped notifications, since they use a somewhat weird, inside-out pagination scheme. If you need to access these in a paginated way, use fetch_next and fetch_previous directly.

Mastodon.**get_pagination_info**(*page:* PaginatableList*[*Entity*]*, *pagination_direction: str*) → PaginationInfo | None

Extracts pagination information from a paginated response.

Returns a PaginationInfo dictionary containing pagination information, or None if not available.

The resulting PaginationInfo is best treated as opaque, though is unlikely to change.

4.15.2 Blurhash decoding

This function allows for easy basic decoding of blurhash strings to images. This requires Mastodon.pys optional "blurhash" feature dependencies.

Mastodon.**decode_blurhash**(*media_dict:* MediaAttachment, *out_size: Tuple[int, int] = (16, 16)*, *size_per_component: bool = True*, *return_linear: bool = True*) → List[List[List[float]]]

Basic media-dict blurhash decoding.

out_size is the desired result size in pixels, either absolute or per blurhash component (this is the default).

By default, this function will return the image as linear RGB, ready for further scaling operations. If you want to display the image directly, set return_linear to False.

Returns the decoded blurhash image as a three-dimensional list: [height][width][3], with the last dimension being RGB colours.

For further info and tips for advanced usage, refer to the documentation for the blurhash module: https://github.com/halcy/blurhash-python

4.15.3 Cache control

Mastodon.**clear_caches**()

Clear cached data for astodon version and streaming base URL. Most programs should not have to call this.

4.15.4 Other utilities

Mastodon.**get_approx_server_time**() → <module 'datetime' from '/home/docs/.asdf/installs/python/3.12.10/lib/python3.12/datetime.py'>

Retrieve the approximate server time

We parse this from the hopefully present "Date" header, but make no effort to compensate for latency.

static Mastodon.**get_status_length**(*text: str*, *spoiler_text: str = ''*) → int

For a given status *text* and *spoiler_text*, return how many characters this status counts as when computing the status length and comparing it against the limit.

Note that there are other limits you may run into, such as the maximum length of a URL, or the maximum length of a usernames domain part. But as long as you do *normal* things, this function will return the correct length for the status text.

4.16 Administration and moderation

These functions allow you to perform moderation actions on users and generally process reports using the API. To do this, you need access to the "admin:read" and/or "admin:write" scopes or their more granular variants (both for the application and the access token), as well as at least moderator access. Mastodon.py will not request these by default, as that would be very dangerous.

BIG WARNING: TREAT ANY ACCESS TOKENS THAT HAVE ADMIN CREDENTIALS AS EXTREMELY, MASSIVELY SENSITIVE DATA AND MAKE EXTRA SURE TO REVOKE THEM AFTER TESTING, NOT LET THEM SIT IN FILES SOMEWHERE, TRACK WHICH ARE ACTIVE, ET CETERA. ANY EXPOSURE OF SUCH ACCESS TOKENS MEANS YOU EXPOSE THE PERSONAL DATA OF ALL YOUR USERS TO WHOEVER HAS THESE TOKENS. TREAT THEM WITH EXTREME CARE.

This is not to say that you should not treat access tokens from admin accounts that do not have admin: scopes attached with a lot of care, but be extra careful with those that do.

4.16.1 Accounts

Mastodon.**admin_accounts_v2**(*origin: str | None = None*, *by_domain: str | None = None*, *status: str | None = None*, *username: str | None = None*, *display_name: str | None = None*, *email: str | None = None*, *ip: str | None = None*, *permissions: str | None = None*, *invited_by:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime = None*, *role_ids: List[str | int | MaybeSnowflakeIdType | datetime] | None = None*, *max_id: str | int | MaybeSnowflakeIdType | datetime | None = None*, *min_id: str | int | MaybeSnowflakeIdType | datetime | None = None*, *since_id: str | int | MaybeSnowflakeIdType | datetime | None = None*, *limit: int | None = None*) → List[*AdminAccount*]

Fetches a list of accounts that match given criteria. By default, local accounts are returned.

- Set *origin* to "local" or "remote" to get only local or remote accounts.
- Set *by_domain* to a domain to get only accounts from that domain.
- Set *status* to one of "active", "pending", "disabled", "silenced" or "suspended" to get only accounts with that moderation status (default: active)
- Set *username* to a string to get only accounts whose username contains this string.
- Set *display_name* to a string to get only accounts whose display name contains this string.
- Set *email* to an email to get only accounts with that email (this only works on local accounts).

- Set *ip* to an ip (as a string, standard v4/v6 notation) to get only accounts whose last active ip is that ip (this only works on local accounts).
- Set *permissions* to "staff" to only get accounts with staff permissions.
- Set *invited_by* to an account id to get only accounts invited by this user.
- Set *role_ids* to a list of role IDs to get only accounts with those roles.

Pagination on this is a bit weird, so I would recommend not doing that and instead manually fetching.

Added: Mastodon v2.9.1, last changed: Mastodon v4.0.0

Mastodon.**admin_accounts**(*remote: bool = False*, *by_domain: str | None = None*, *status: str = 'active'*, *username: str | None = None*, *display_name: str | None = None*, *email: str | None = None*, *ip: str | None = None*, *staff_only: bool = False*, *max_id: str | int | MaybeSnowflakeIdType | datetime | None = None*, *min_id: str | int | MaybeSnowflakeIdType | datetime | None = None*, *since_id: str | int | MaybeSnowflakeIdType | datetime | None = None*, *limit: int | None = None*)

Currently a synonym for admin_accounts_v1, now deprecated. You are strongly encouraged to use admin_accounts_v2 instead, since this one is kind of bad.

!!!!! This function may be switched to calling the v2 API in the future. This is your warning. If you want to keep using v1, use it explicitly. !!!!!

Pagination on this is a bit weird, so I would recommend not doing that and instead manually fetching.

Added: Mastodon v2.9.1, last changed: Mastodon v2.9.1

Mastodon.**admin_accounts_v1**(*remote: bool = False*, *by_domain: str | None = None*, *status: str = 'active'*, *username: str | None = None*, *display_name: str | None = None*, *email: str | None = None*, *ip: str | None = None*, *staff_only: bool = False*, *max_id: str | int | MaybeSnowflakeIdType | datetime | None = None*, *min_id: str | int | MaybeSnowflakeIdType | datetime | None = None*, *since_id: str | int | MaybeSnowflakeIdType | datetime | None = None*, *limit: int | None = None*) → *AdminAccount*

Fetches a list of accounts that match given criteria. By default, local accounts are returned.

- Set *remote* to True to get remote accounts, otherwise local accounts are returned (default: local accounts)
- Set *by_domain* to a domain to get only accounts from that domain.
- Set *status* to one of "active", "pending", "disabled", "silenced" or "suspended" to get only accounts with that moderation status (default: active)
- Set *username* to a string to get only accounts whose username contains this string.
- Set *display_name* to a string to get only accounts whose display name contains this string.
- Set *email* to an email to get only accounts with that email (this only works on local accounts).
- Set *ip* to an ip (as a string, standard v4/v6 notation) to get only accounts whose last active ip is that ip (this only works on local accounts).
- Set *staff_only* to True to only get staff accounts (this only works on local accounts).

Note that setting the boolean parameters to False does not mean "give me users to which this does not apply" but instead means "I do not care if users have this attribute".

Deprecated in Mastodon version 3.5.0.

Pagination on this is a bit weird, so I would recommend not doing that and instead manually fetching.

Added: Mastodon v2.9.1, last changed: Mastodon v2.9.1 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**admin_account**(*id:* Account | AdminAccount | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *AdminAccount*

Fetches a single admin account for the user with the given id.

Added: Mastodon v2.9.1, last changed: Mastodon v2.9.1 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**admin_account_enable**(*id:* Account | AdminAccount | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *AdminAccount*

Reenables login for a local account for which login has been disabled.

The returned object reflects the updates to the account.

Added: Mastodon v2.9.1, last changed: Mastodon v2.9.1 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**admin_account_approve**(*id:* Account | AdminAccount | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *AdminAccount*

Approves a pending account.

The returned object reflects the updates to the account.

Added: Mastodon v2.9.1, last changed: Mastodon v2.9.1 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**admin_account_reject**(*id:* Account | AdminAccount | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *AdminAccount*

Rejects and deletes a pending account.

The returned object is that of the now-deleted account.

Added: Mastodon v2.9.1, last changed: Mastodon v2.9.1 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**admin_account_unsilence**(*id:* Account | AdminAccount | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *AdminAccount*

Unsilences an account.

The returned object reflects the updates to the account.

Added: Mastodon v2.9.1, last changed: Mastodon v2.9.1 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**admin_account_unsuspend**(*id:* Account | AdminAccount | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *AdminAccount*

Unsuspends an account.

The returned object reflects the updates to the account.

Added: Mastodon v2.9.1, last changed: Mastodon v2.9.1 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**admin_account_moderate**(*id:* Account | AdminAccount | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *action: str* | *None* = *None*, *report_id:* AdminReport | *str* | *int* | *None* = *None*, *warning_preset_id: str* | *int* | *None* = *None*, *text: str* | *None* = *None*, *send_email_notification: bool* | *None* = *True*)

Perform a moderation action on an account.

Valid actions are:

- "disable" - for a local user, disable login.
- "silence" - hide the users posts from all public timelines.
- "suspend" - irreversibly delete all the user's posts, past and future.
- "sensitive" - forcce an accounts media visibility to always be sensitive.

If no action is specified, the user is only issued a warning.

Specify the id of a report as *report_id* to close the report with this moderation action as the resolution. Specify *warning_preset_id* to use a warning preset as the notification text to the user, or *text* to specify text directly. If both are specified, they are concatenated (preset first). Note that there is currently no API to retrieve or create warning presets.

Set *send_email_notification* to False to not send the user an email notification informing them of the moderation action.

Added: Mastodon v2.9.1, last changed: Mastodon v2.9.1

4.16.2 Reports

`Mastodon.admin_reports`(*resolved: bool | None = False*, *account_id:* Account | AdminAccount | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *target_account_id:* Account | AdminAccount | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *max_id: str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *min_id: str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *since_id: str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *limit: int* | *None = None*) → *PaginatableList*[*AdminReport*]

Fetches the list of reports.

Set *resolved* to True to search for resolved reports. *account_id* and *target_account_id* can be used to get reports filed by or about a specific user.

Added: Mastodon v2.9.1, last changed: Mastodon v2.9.1

`Mastodon.admin_report`(*id:* AdminReport | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *AdminReport*

Fetches the report with the given id.

Added: Mastodon v2.9.1, last changed: Mastodon v2.9.1 (parameters), Mastodon v4.0.0 (return value)

`Mastodon.admin_report_assign`(*id:* AdminReport | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *AdminReport*

Assigns the given report to the logged-in user.

The returned object reflects the updates to the report.

Added: Mastodon v2.9.1, last changed: Mastodon v2.9.1 (parameters), Mastodon v4.0.0 (return value)

`Mastodon.admin_report_unassign`(*id:* AdminReport | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *AdminReport*

Unassigns the given report from the logged-in user.

The returned object reflects the updates to the report.

Added: Mastodon v2.9.1, last changed: Mastodon v2.9.1 (parameters), Mastodon v4.0.0 (return value)

`Mastodon.admin_report_reopen`(*id:* AdminReport | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *AdminReport*

Reopens a closed report.

The returned object reflects the updates to the report.

Added: Mastodon v2.9.1, last changed: Mastodon v2.9.1 (parameters), Mastodon v4.0.0 (return value)

`Mastodon.admin_report_resolve`(*id:* AdminReport | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *AdminReport*

Marks a report as resolved (without taking any action).

The returned object reflects the updates to the report.

Added: Mastodon v2.9.1, last changed: Mastodon v2.9.1 (parameters), Mastodon v4.0.0 (return value)

4.16.3 Federation

Mastodon.**admin_domain_blocks**(*id: str | int | MaybeSnowflakeIdType | datetime | None = None*, *max_id: str | int | MaybeSnowflakeIdType | datetime | None = None*, *min_id: str | int | MaybeSnowflakeIdType | datetime | None = None*, *since_id: str | int | MaybeSnowflakeIdType | datetime | None = None*, *limit: int | None = None*) → *AdminDomainBlock | PaginatableList*[*AdminDomainBlock*]

Fetches a list of blocked domains. Requires scope *admin:read:domain_blocks*.

Provide an *id* to fetch a specific domain block based on its database id.

Raises a *MastodonAPIError* if the specified block does not exist.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0

Mastodon.**admin_create_domain_block**(*domain: str*, *severity: str | None = None*, *reject_media: bool | None = None*, *reject_reports: bool | None = None*, *private_comment: str | None = None*, *public_comment: str | None = None*, *obfuscate: bool | None = None*) → *AdminDomainBlock*

Perform a moderation action on a domain. Requires scope *admin:write:domain_blocks*.

Valid severities are:

- "silence" - hide all posts from federated timelines and do not show notifications to local users from the remote instance's users unless they are following the remote user.
- "suspend" - deny interactions with this instance going forward. This action is reversible.
- "limit" - generally used with reject_media=true to force reject media from an instance without silencing or suspending..

If no action is specified, the domain is only silenced. *domain* is the domain to block. Note that using the top level domain will also imapct all subdomains. ie, example.com will also impact subdomain.example.com. *reject_media* will not download remote media on to your local instance media storage. *reject_reports* ignores all reports from the remote instance. *private_comment* sets a private admin comment for the domain. *public_comment* sets a publicly available comment for this domain, which will be available to local users and may be available to everyone depending on your settings. *obfuscate* censors some part of the domain name. Useful if the domain name contains unwanted words like slurs.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0 (parameters), Mastodon v4.3.0 (return value)

Mastodon.**admin_update_domain_block**(*id*, *severity: str | None = None*, *reject_media: bool | None = None*, *reject_reports: bool | None = None*, *private_comment: str | None = None*, *public_comment: str | None = None*, *obfuscate: bool | None = None*) → *AdminDomainBlock*

Modify existing moderation action on a domain. Requires scope *admin:write:domain_blocks*.

Valid severities are:

- "silence" - hide all posts from federated timelines and do not show notifications to local users from the remote instance's users unless they are following the remote user.
- "suspend" - deny interactions with this instance going forward. This action is reversible.
- "limit" - generally used with reject_media=true to force reject media from an instance without silencing or suspending.

If no action is specified, the domain is only silenced. *domain* is the domain to block. Note that using the top level domain will also imapct all subdomains. ie, example.com will also impact subdomain.example.com. *reject_media* will not download remote media on to your local instance media storage. *reject_reports* ignores all reports from the remote instance. *private_comment* sets a private admin comment for the domain. *public_comment* sets a publicly available comment for this domain, which will be available to local users and may be available to everyone depending on your settings. *obfuscate* censors some part of the domain name. Useful if the domain name contains unwanted words like slurs.

Raises a *MastodonAPIError* if the specified block does not exist.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0 (parameters), Mastodon v4.3.0 (return value)

Mastodon.**admin_delete_domain_block**(*id=typing.Union[mastodon.return_types.AdminDomainBlock, str, int, mastodon.types_base.MaybeSnowflakeIdType, datetime.datetime]*)

Removes moderation action against a given domain. Requires scope *admin:write:domain_blocks*.

Provide an *id* to remove a specific domain block based on its database id.

Raises a *MastodonAPIError* if the specified block does not exist.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0

4.16.4 Moderation actions

Mastodon.**admin_measures**(*start_at*, *end_at*, *active_users: bool = False*, *new_users: bool = False*, *interactions: bool = False*, *opened_reports: bool = False*, *resolved_reports: bool = False*, *tag_accounts:* Tag | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *tag_uses:* Tag | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *tag_servers:* Tag | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *instance_accounts: str* | *None = None*, *instance_media_attachments: str* | *None = None*, *instance_reports: str* | *None = None*, *instance_statuses: str* | *None = None*, *instance_follows: str* | *None = None*, *instance_followers: str* | *None = None*) → *NonPaginatableList*[*AdminMeasure*]

Retrieves numerical instance information for the time period (at day granularity) between *start_at* and *end_at*.

- *active_users*: Pass true to retrieve the number of active users on your instance within the time period
- *new_users*: Pass true to retrieve the number of users who joined your instance within the time period
- *interactions*: Pass true to retrieve the number of interactions (favourites, boosts, replies) on local statuses within the time period
- *opened_reports*: Pass true to retrieve the number of reports filed within the time period
- *resolved_reports* = Pass true to retrieve the number of reports resolved within the time period
- *tag_accounts*: Pass a tag ID to get the number of accounts which used that tag in at least one status within the time period
- *tag_uses*: Pass a tag ID to get the number of statuses which used that tag within the time period
- *tag_servers*: Pass a tag ID to to get the number of remote origin servers for statuses which used that tag within the time period
- *instance_accounts*: Pass a domain to get the number of accounts originating from that remote domain within the time period
- *instance_media_attachments*: Pass a domain to get the amount of space used by media attachments from that remote domain within the time period
- *instance_reports*: Pass a domain to get the number of reports filed against accounts from that remote domain within the time period

- *instance_statuses*: Pass a domain to get the number of statuses originating from that remote domain within the time period
- *instance_follows*: Pass a domain to get the number of accounts from a remote domain followed by that local user within the time period
- *instance_followers*: Pass a domain to get the number of local accounts followed by accounts from that remote domain within the time period

This API call is relatively expensive - watch your servers load if you want to get a lot of statistical data. Especially the instance_statuses stats might take a long time to compute and, in fact, time out.

There is currently no way to get tag IDs implemented in Mastodon.py, because the Mastodon public API does not implement one. This will be fixed in a future release.

Added: Mastodon v3.5.0, last changed: Mastodon v3.5.0

Mastodon.**admin_dimensions**(*start_at: datetime*, *end_at: datetime*, *limit: int | None = None*, *languages: bool = False*, *sources: bool = False*, *servers: bool = False*, *space_usage: bool = False*, *software_versions: bool = False*, *tag_servers:* Tag | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *tag_languages:* Tag | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *instance_accounts: str | None = None*, *instance_languages: str | None = None*) → *NonPaginatableList*[*AdminDimension*]

Retrieves primarily categorical instance information for the time period (at day granularity) between *start_at* and *end_at*.

- *languages*: Pass true to get the most-used languages on this server
- *sources*: Pass true to get the most-used client apps on this server
- *servers*: Pass true to get the remote servers with the most statuses
- *space_usage*: Pass true to get the how much space is used by different components your software stack
- *software_versions*: Pass true to get the version numbers for your software stack
- *tag_servers*: Pass a tag ID to get the most-common servers for statuses including a trending tag
- *tag_languages*: Pass a tag ID to get the most-used languages for statuses including a trending tag
- *instance_accounts*: Pass a domain to get the most-followed accounts from a remote server
- *instance_languages*: Pass a domain to get the most-used languages from a remote server

Pass *limit* to set how many results you want on queries where that makes sense.

This API call is relatively expensive - watch your servers load if you want to get a lot of statistical data.

There is currently no way to get tag IDs implemented in Mastodon.py, because the Mastodon public API does not implement one. This will be fixed in a future release.

Added: Mastodon v3.5.0, last changed: Mastodon v3.5.0

Mastodon.**admin_retention**(*start_at: datetime*, *end_at: datetime*, *frequency: str = 'day'*) → *NonPaginatableList*[*AdminRetention*]

Gets user retention statistics (at *frequency* - “day” or “month” - granularity) between *start_at* and *end_at*.

Added: Mastodon v3.5.0, last changed: Mastodon v3.5.0

4.16.5 Canonical email blocks

Mastodon.**admin_canonical_email_blocks**(*max_id: str | int | MaybeSnowflakeIdType | datetime | None = None*, *min_id: str | int | MaybeSnowflakeIdType | datetime | None = None*, *since_id: str | int | MaybeSnowflakeIdType | datetime | None = None*, *limit: int | None = None*) → *PaginatableList*[*AdminCanonicalEmailBlock*]

Fetches a list of canonical email blocks. Requires scope *admin:read:canonical_email_blocks*.

The returned list may be paginated using max_id, min_id, and since_id.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0

Mastodon.**admin_canonical_email_block**(*id: str | int | MaybeSnowflakeIdType | datetime*) → *AdminCanonicalEmailBlock*

Fetch a single canonical email block by ID. Requires scope *admin:read:canonical_email_blocks*.

Raises *MastodonAPIError* if the email block does not exist.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**admin_test_canonical_email_block**(*email: str*) → *NonPaginatableList*[*AdminCanonicalEmailBlock*]

Canonicalize and hash an email address, returning all matching canonical email blocks. Requires scope *admin:read:canonical_email_blocks*.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0

Mastodon.**admin_create_canonical_email_block**(*email: str | None = None*, *canonical_email_hash: str | None = None*) → *AdminCanonicalEmailBlock*

Block a canonical email. Requires scope *admin:write:canonical_email_blocks*.

Either *email* (which will be canonicalized and hashed) or *canonical_email_hash* must be provided.

Up do date details about the canonicalization and hashing process can be found here:

https://github.com/mastodon/mastodon/blob/main/app/helpers/email_helper.rb

As of Mastodon v4.4.0:

- Everything lowercased
- Dots are removed from the part before the @
- Anything after a + is removed
- The hash in use is SHA256

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**admin_delete_canonical_email_block**(*id: str | int | MaybeSnowflakeIdType | datetime*) → *AdminCanonicalEmailBlock*

Delete a canonical email block by ID. Requires scope *admin:write:canonical_email_blocks*.

Raises *MastodonAPIError* if the email block does not exist.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0 (parameters), Mastodon v4.0.0 (return value)

4.16.6 Email domain blocks

Mastodon.**admin_email_domain_blocks**(*max_id: str | int | MaybeSnowflakeIdType | datetime | None = None*, *min_id: str | int | MaybeSnowflakeIdType | datetime | None = None*, *since_id: str | int | MaybeSnowflakeIdType | datetime | None = None*, *limit: int | None = None*) → *PaginatableList*[*AdminEmailDomainBlock*]

Fetches a list of blocked email domains. Requires scope *admin:read:email_domain_blocks*.

The returned list may be paginated using max_id, min_id, and since_id.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0

Mastodon.**admin_email_domain_block**(*id: str | int | MaybeSnowflakeIdType | datetime*) → *AdminEmailDomainBlock*

Fetch a single blocked email domain by ID. Requires scope *admin:read:email_domain_blocks*.

Raises *MastodonAPIError* if the email domain block does not exist.

Added: Mastodon v4.1.0, last changed: Mastodon v4.1.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**admin_create_email_domain_block**(*domain: str*) → *AdminEmailDomainBlock*

Block an email domain from signups. Requires scope *admin:write:email_domain_blocks*.

If the domain contains invalid characters, a *MastodonAPIError* will be raised.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**admin_delete_email_domain_block**(*id: str | int | MaybeSnowflakeIdType | datetime*)

Remove an email domain block. Requires scope *admin:write:email_domain_blocks*.

Raises *MastodonAPIError* if the email domain block does not exist.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0

4.16.7 IP blocks

Mastodon.**admin_ip_blocks**(*max_id: str | int | MaybeSnowflakeIdType | datetime | None = None*, *min_id: str | int | MaybeSnowflakeIdType | datetime | None = None*, *since_id: str | int | MaybeSnowflakeIdType | datetime | None = None*, *limit: int | None = None*) → *PaginatableList*[*AdminIpBlock*]

Fetches a list of blocked IP addresses and ranges. Requires scope *admin:read:ip_blocks*.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0

Mastodon.**admin_ip_block**(*id:* AdminIpBlock *| str | int | MaybeSnowflakeIdType | datetime*) → *AdminIpBlock*

Fetch a single blocked IP address or range by ID. Requires scope *admin:read:ip_blocks*.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**admin_create_ip_block**(*ip: str*, *severity: str*, *comment: str | None = None*, *expires_in: int | None = None*) → *AdminIpBlock*

Block an IP address or range from signups. Requires scope *admin:write:ip_blocks*.

Provide the IP address as a CIDR range, e.g. "192.168.1.1/32" to block just that IP address, or "8.8.8.8/24" to block all addresses in the 8.8.8.* subnet.

severity can be one of three values:

- "sign_up_requires_approval" - signups from this IP will require manual approval
- "sign_up_block" - signups from this IP will be blocked

- "no_access" - all access from this IP will be blocked

expires_in is the number of seconds until the block expires. If not provided, the block will be permanent.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**admin_update_ip_block**(*id:* AdminIpBlock | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *ip: str* | *None = None*, *severity: str* | *None = None*, *comment: str* | *None = None*, *expires_in: int* | *None = None*) → *AdminIpBlock*

Update an existing IP block. Requires scope *admin:write:ip_blocks*.

expires_in is the number of seconds until the block expires. If not provided, the block will be permanent.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**admin_delete_ip_block**(*id:* AdminIpBlock | *str* | *int* | MaybeSnowflakeIdType | *datetime*)

Remove an IP block. Requires scope *admin:write:ip_blocks*.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0

4.16.8 Trend management

Mastodon.**admin_trending_tags**(*limit: int* | *None = None*) → *NonPaginatableList*[*Tag*]

Admin version of *trending_tags()*. Includes unapproved tags.

The returned list is sorted, descending, by the instance's trending algorithm.

Returns a regular Tag without admin attributes between Mastodon.py v4.0.0 and v4.1.0 due to a bug.

Added: Mastodon v3.5.0, last changed: Mastodon v3.5.0

Mastodon.**admin_trending_statuses**() → *NonPaginatableList*[*Status*]

Admin version of *trending_statuses()*. Includes unapproved tags.

The returned list is sorted, descending, by the instance's trending algorithm.

Added: Mastodon v3.5.0, last changed: Mastodon v3.5.0

Mastodon.**admin_trending_links**() → *NonPaginatableList*[*PreviewCard*]

Admin version of *trending_links()*. Includes unapproved tags.

The returned list is sorted, descending, by the instance's trending algorithm.

Added: Mastodon v3.5.0, last changed: Mastodon v3.5.0

Mastodon.**admin_approve_trending_link**(*id:* PreviewCard | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *PreviewCard*

Approve a trending link. Requires scope *admin:write*.

Added: Mastodon v4.2.0, last changed: Mastodon v4.2.0 (parameters), Mastodon v4.3.0 (return value)

Mastodon.**admin_reject_trending_link**(*id:* PreviewCard | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *PreviewCard*

Reject a trending link. Requires scope *admin:write*.

Added: Mastodon v4.2.0, last changed: Mastodon v4.2.0 (parameters), Mastodon v4.3.0 (return value)

Mastodon.**admin_approve_trending_status**(*id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Status*

Approve a trending status. Requires scope *admin:write*.

Added: Mastodon v4.2.0, last changed: Mastodon v4.2.0 (parameters), Mastodon v4.4.0 (return value)

Mastodon.**admin_reject_trending_status**(*id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Status*

Reject a trending status. Requires scope *admin:write*.

Added: Mastodon v4.2.0, last changed: Mastodon v4.2.0 (parameters), Mastodon v4.4.0 (return value)

Mastodon.**admin_approve_trending_tag**(*id:* Tag | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Tag*

Approve a trending tag. Requires scope *admin:write*.

Added: Mastodon v4.2.0, last changed: Mastodon v4.2.0 (parameters), Mastodon v4.4.0 (return value)

Mastodon.**admin_reject_trending_tag**(*id:* Tag | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Tag*

Reject a trending tag. Requires scope *admin:write*.

Added: Mastodon v4.2.0, last changed: Mastodon v4.2.0 (parameters), Mastodon v4.4.0 (return value)

4.17 Contributing

4.17.1 How to contribute

Mastodon.py is incomplete a lot of the time because Mastodon has a very rich API with many functions, not all of which are implemented here. Even when it is complete for a given Mastodon API version, there are forks and other Mastodon-API-compatible software that implement their own methods which Mastodon.py could in principle support. And even when all of that work is done, it will inevitably have bugs, or places where the library could be made easier to use (which, really, are also bugs), missing tests that could catch bugs quicker, tooling to make updating everything faster, et cetera.

You can help get more of this done, and you should! This can take many forms: If you notice somtehing is missing, broken or confusing:

- You could file an issue on github, either with or without suggestions for how to fix the issue: https://github.com/halcy/Mastodon.py/issues
- You could, after filing an issue, do a PR that fixes that issue
- You could even just vaguely complain in my (https://icosahedron.website/@halcy) general direction on Mastodon

All of these help immensely, even if it's just "hey, I don't really get why X isn't working". We can't make the library better if we don't know what the actual issues people have are, so while I'm not going to implement every suggestion and do have some ideas of what does and does not make a good library, your feedback is, in fact, extremely valuable and welcome.

If you're looking for some "starter issues" to address: Currently, we don't have support for much of any of the new 4.0.0 API endpoints implemented. Pick one and have a go, especially from the admin API. Tests are somewhat annoying to set up, as they need to run against a live mastodon instance - great if you can write them, but feel free to skip out on them, too, or just write them "in the dry" without actually running them and leaving that for someone else.

4.17.2 Tests

Mastodon.py has an extensive suite of tests. The purpose of these is twofold:

- Make sure nothing is broken and that there aren't any regressions
- Where the official docs are unclear, verify assumptions we make about the Mastodon API and document the results

The tests use pytest and pytest-recording so that they can be ran even without a mastodon server, but new tests require setting up a mastodon dev server. Further documentation can be found in the "tests" directory in the repository.

4.18 Every function on a huge CTRL-F-able page

Mastodon.**retrieve_mastodon_version**() → str

Determine installed Mastodon version and set major, minor and patch (not including RC info) accordingly.

Returns the version string, possibly including rc info.

Mastodon.**verify_minimum_version**(*version_str: str*, *cached: bool = False*) → bool

Update version info from server and verify that at least the specified version is present.

If you specify "cached", the version info update part is skipped.

Returns True if version requirement is satisfied, False if not.

class mastodon.types_base.**AttribAccessDict**(***kwargs*)

Base return object class for Mastodon.py.

Inherits from dict, but allows access via attributes as well as if it was a dataclass.

While the subclasses implement specific fields with proper typing, parsing and documentation, they all inherit from this class, and parsing is extremely permissive to allow for forward and backward compatibility as well as compatibility with other implementations of the Mastodon API.

This class can ALSO have pagination information attached, for paginating lists *inside* the object, because that's what Mastodon 4.3.0 does for groupee notifications. This is special cased in the class definition, though.

class mastodon.types_base.**PaginatableList**(**args*, ***kwargs*)

This is a list with pagination information attached.

It is returned by the API when a list of items is requested, and the response contains a Link header with pagination information.

class mastodon.types_base.**NonPaginatableList**(**args*, ***kwargs*)

This is just a list, without pagination information but annotated for serialization and deserialization.

class mastodon.types_base.**MaybeSnowflakeIdType**(*value*, **args*, ***kwargs*)

Represents, maybe, a snowflake ID.

Contains what a regular ID can contain (int or str) and will convert to int if containing an int or a str that naturally converts to an int (e.g. "123").

Can *also* contain a *datetime* which gets converted to a timestamp.

It's also just *maybe* a snowflake ID, because some implementations may not use those.

This may seem annoyingly complex, but the goal here is: 1) If constructed with some ID, return that ID unchanged, so equality and hashing work 2) Allow casting to int and str, just like a regular ID 3) Halfway transparently convert to and from datetime with the correct format for the server we're talking to

mastodon.types_base.**IdType**

IDs returned from Mastodon.py ar either primitive (int or str) or snowflake (still int or str, but potentially convertible to datetime), but also a datetime (which will get converted to a snowflake id).

class mastodon.types_base.**Entity**

Base class for everything returned by the API. This is a union of *AttribAccessDict* and *EntityList*.

Defines two methods: to_json(), and (static) from_json(), for serializing and deserializing to/from JSON.

mastodon.types_base.**EntityList**

Lists in Mastodon.py are either regular or paginatable, so this is a union of *NonPaginatableList* and *PaginatableList*.

class mastodon.return_types.**Account**(***kwargs*)

A user acccount, local or remote.

Example:

```
# Returns a Account object
mastodon.account(<account id>)
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Account/

class mastodon.return_types.**AccountField**(***kwargs*)

A field, displayed on a users profile (e.g. "Pronouns", "Favorite color").

Example:

```
# Returns a AccountField object
mastodon.account(<account id>).fields[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Account/

class mastodon.return_types.**Role**(***kwargs*)

A role granting a user a set of permissions.

Example:

```
# Returns a Role object
mastodon.account_verify_credentials().role
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Role/

class mastodon.return_types.**CredentialAccountSource**(***kwargs*)

Source values useful for editing a user's profile.

Example:

```
# Returns a CredentialAccountSource object
mastodon.account_verify_credentials()["source"]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Account/

class mastodon.return_types.**Status**(***kwargs*)

A single status / toot / post.

Example:

```
# Returns a Status object
mastodon.toot("Hello from Python")
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Status/

class mastodon.return_types.**StatusEdit**(***kwargs*)

An object representing a past version of an edited status.

Example:

```
# Returns a StatusEdit object
mastodon.status_history(<status id>)[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/StatusEdit/

class mastodon.return_types.**FilterResult**(***kwargs*)

A filter action that should be taken on a status.

Example:

```
# Returns a FilterResult object
mastodon.status(<status id>).filtered[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/FilterResult/

class mastodon.return_types.**StatusMention**(***kwargs*)

A mention of a user within a status.

Example:

```
# Returns a StatusMention object
mastodon.toot("@admin he doing it sideways").mentions[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Mention/

class mastodon.return_types.**ScheduledStatus**(***kwargs*)

A scheduled status / toot to be eventually posted.

Example:

```
# Returns a ScheduledStatus object
mastodon.status_post("futureposting", scheduled_at=the_future)
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/ScheduledStatus/

class mastodon.return_types.**ScheduledStatusParams**(***kwargs*)

Parameters for a status / toot to be posted in the future.

Example:

```
# Returns a ScheduledStatusParams object
mastodon.status_post("futureposting... 2", scheduled_at=the_future).params
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/ScheduledStatus/

class mastodon.return_types.**Poll**(***kwargs*)

A poll attached to a status.

Example:

```
# Returns a Poll object
mastodon.poll(<poll id>)
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Poll/

class mastodon.return_types.**PollOption**(***kwargs*)

A poll option within a poll.

Example:

```
# Returns a PollOption object
mastodon.poll(<poll id>).options[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Poll/

class mastodon.return_types.**Conversation**(***kwargs*)

A conversation (using direct / mentions-only visibility) between two or more users.

Example:

```
# Returns a Conversation object
mastodon.conversations()[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Conversation/

class mastodon.return_types.**Tag**(***kwargs*)

A hashtag, as part of a status or on its own (e.g. trending).

Example:

```
# Returns a Tag object
mastodon.trending_tags()[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Tag/

class mastodon.return_types.**TagHistory**(***kwargs*)

Usage history for a hashtag.

Example:

```
# Returns a TagHistory object
mastodon.trending_tags()[0].history[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Tag/

class mastodon.return_types.**CustomEmoji**(***kwargs*)

A custom emoji.

Example:

```
# Returns a CustomEmoji object
mastodon.toot(":sidekiqin:").emojis[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/CustomEmoji/

class mastodon.return_types.**Application**(***kwargs*)

Information about an app (in terms of the API).

Example:

```
# Returns a Application object
mastodon.app_verify_credentials()
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Application/

class mastodon.return_types.**Relationship**(***kwargs*)

Information about the relationship between two users.

Example:

```
# Returns a Relationship object
mastodon.account_relationships(<account id>)[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Relationship/

class mastodon.return_types.**FilterV2**(***kwargs*)

Information about a keyword / status filter.

Example:

```
# Returns a FilterV2 object
mastodon.filters_v2()[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Filter/

class mastodon.return_types.**Notification**(***kwargs*)

A notification about some event, like a new reply or follower.

Example:

```
# Returns a Notification object
mastodon.notifications()[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Notification/

class mastodon.return_types.**Context**(***kwargs*)

The conversation context for a given status, i.e. its predecessors (that it replies to) and successors (that reply to it).

Example:

```
# Returns a Context object
mastodon.status_context(<status id>)
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Context/

class mastodon.return_types.**UserList**(***kwargs*)

A list of users.

Example:

```
# Returns a UserList object
mastodon.lists()[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/List/

class mastodon.return_types.**MediaAttachment**(***kwargs*)

A piece of media (like an image, video, or audio file) that can be or has been attached to a status.

Example:

```
# Returns a MediaAttachment object
mastodon.media_post("image.jpg", "image/jpeg")["meta"]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/MediaAttachment/

class mastodon.return_types.**MediaAttachmentMetadataContainer**(***kwargs*)

An object holding metadata about a media attachment and its thumbnail. In addition to the documented fields, there may be additional fields. These are not documented, not guaranteed to be present (they are a Mastodon implementation detail), and may change without notice, so relying on them is not recommended.

Example:

```
# Returns a MediaAttachmentMetadataContainer object
mastodon.media_post("audio.mp3").meta
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/MediaAttachment/

class mastodon.return_types.**MediaAttachmentImageMetadata**(***kwargs*)

Metadata for an image media attachment.

Example:

```
# Returns a MediaAttachmentImageMetadata object
mastodon.media_post("image.jpg").meta.original
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/MediaAttachment/

class mastodon.return_types.**MediaAttachmentVideoMetadata**(***kwargs*)

Metadata for a video attachment. This can be a proper video, or a gifv (a looping, soundless animation). Both use the same data model currently, though there is a possibility that they could be split in the future.

Example:

```
# Returns a MediaAttachmentVideoMetadata object
mastodon.media_post("video.mp4").meta.original
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/MediaAttachment/

class mastodon.return_types.**MediaAttachmentAudioMetadata**(***kwargs*)

Metadata for an audio media attachment.

Example:

```
# Returns a MediaAttachmentAudioMetadata object
mastodon.media_post("audio.mp3").meta.original
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/MediaAttachment/

class mastodon.return_types.**MediaAttachmentFocusPoint**(***kwargs*)

The focus point for a media attachment, for cropping purposes.

Example:

```
# Returns a MediaAttachmentFocusPoint object
mastodon.media_post("image.jpg").meta.focus
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/MediaAttachment/

class mastodon.return_types.**MediaAttachmentColors**(***kwargs*)

Object describing the accent colors for a media attachment.

Example:

```
# Returns a MediaAttachmentColors object
mastodon.media_post("image.jpg").meta.colors
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/MediaAttachment/

class mastodon.return_types.**PreviewCard**(***kwargs*)

A preview card attached to a status, e.g. for an embedded video or link.

Example:

```
# Returns a PreviewCard object
mastodon.status_card(<status id>)
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/PreviewCard/

class mastodon.return_types.**TrendingLinkHistory**(***kwargs*)

A history entry for a trending link.

Example:

```
# Returns a TrendingLinkHistory object
mastodon.trending_links()[0].history[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/PreviewCard/#trends-link

class mastodon.return_types.**PreviewCardAuthor**(***kwargs*)

A preview card attached to a status, e.g. for an embedded video or link.

Example:

```
# Returns a PreviewCardAuthor object
mastodon.status_card(<status id>).authors[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/PreviewCardAuthor/

class mastodon.return_types.**SearchV2**(***kwargs*)

A search result, with accounts, hashtags and statuses.

Example:

```
# Returns a SearchV2 object
mastodon.search("<search query>")
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Search/

class mastodon.return_types.**Instance**(***kwargs*)

Information about an instance. V1 API version.

Example:

```
# Returns a Instance object
mastodon.instance_v1()
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/V1_Instance/

class mastodon.return_types.**InstanceConfiguration**(***kwargs*)

Configuration values for this instance, especially limits and enabled features.

Example:

```
# Returns a InstanceConfiguration object
mastodon.instance_v1().configuration
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/methods/instance/

class mastodon.return_types.**InstanceURLs**(***kwargs*)

A list of URLs related to an instance.

Example:

```
# Returns a InstanceURLs object
mastodon.instance_v1().urls
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/V1_Instance/

class mastodon.return_types.**InstanceV2**(***kwargs*)

Information about an instance.

Example:

```
# Returns a InstanceV2 object
mastodon.instance_v2()
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Instance/

class mastodon.return_types.**InstanceIcon**(***kwargs*)

Icon for the instance, in a specific size.

Example:

```
# Returns a InstanceIcon object
mastodon.instance_v2().icon[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/methods/instance/#InstanceIcon

class mastodon.return_types.**InstanceConfigurationV2**(***kwargs*)

Configuration values for this instance, especially limits and enabled features.

Example:

```
# Returns a InstanceConfigurationV2 object
mastodon.instance_v2().configuration
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/methods/instance/

class mastodon.return_types.**InstanceVapidKey**(***kwargs*)

The VAPID key used by this instance to sign webpush requests.

Example:

```
# Returns a InstanceVapidKey object
mastodon.instance_v2().configuration.vapid
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/methods/instance/

class mastodon.return_types.**InstanceURLsV2**(***kwargs*)

A list of URLs related to an instance.

Example:

```
# Returns a InstanceURLsV2 object
mastodon.instance_v2().configuration.urls
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Instance/

class mastodon.return_types.**InstanceThumbnail**(***kwargs*)

Extended information about an instances thumbnail.

Example:

```
# Returns a InstanceThumbnail object
mastodon.instance().thumbnail
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/V1_Instance/

class mastodon.return_types.**InstanceThumbnailVersions**(***kwargs*)

Different resolution versions of the image representing the instance.

Example:

```
# Returns a InstanceThumbnailVersions object
mastodon.instance().thumbnail.versions
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Instance/

class mastodon.return_types.**InstanceStatistics**(***kwargs*)

Usage statistics for an instance.

Example:

```
# Returns a InstanceStatistics object
mastodon.instance_v1().stats
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Instance/

class mastodon.return_types.**InstanceUsage**(***kwargs*)

Usage / recent activity information for this instance.

Example:

```
# Returns a InstanceUsage object
mastodon.instance().usage
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Instance/

class mastodon.return_types.**InstanceUsageUsers**(***kwargs*)

Recent active user information about this instance.

Example:

```
# Returns a InstanceUsageUsers object
mastodon.instance().usage.users
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Instance/

class mastodon.return_types.**Rule**(***kwargs*)

A rule that instance staff has specified users must follow on this instance.

Example:

```
# Returns a Rule object
mastodon.instance().rules[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Rule/

class mastodon.return_types.**InstanceRegistrations**(***kwargs*)

Registration information for this instance, like whether registrations are open and whether they require approval.

Example:

```
# Returns a InstanceRegistrations object
mastodon.instance_v2().registrations
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Instance/

class mastodon.return_types.**InstanceContact**(***kwargs*)

Contact information for this instances' staff.

Example:

```
# Returns a InstanceContact object
mastodon.instance().contact
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Instance/

class mastodon.return_types.**InstanceAccountConfiguration**(***kwargs*)

Configuration values relating to accounts.

Example:

```
# Returns a InstanceAccountConfiguration object
mastodon.instance().configuration.accounts
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/methods/instance/

class mastodon.return_types.**InstanceStatusConfiguration**(***kwargs*)

Configuration values relating to statuses.

Example:

```
# Returns a InstanceStatusConfiguration object
mastodon.instance().configuration.statuses
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/methods/instance/

class mastodon.return_types.**InstanceTranslationConfiguration**(***kwargs*)

Configuration values relating to translation.

Example:

```
# Returns a InstanceTranslationConfiguration object
mastodon.instance_v2().configuration.translation
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/methods/instance/

class mastodon.return_types.**InstanceMediaConfiguration**(***kwargs*)

Configuration values relating to media attachments.

Example:

```
# Returns a InstanceMediaConfiguration object
mastodon.instance().configuration.media_attachments
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/methods/instance/

class mastodon.return_types.**InstancePollConfiguration**(***kwargs*)

Configuration values relating to polls.

Example:

```
# Returns a InstancePollConfiguration object
mastodon.instance().configuration.polls
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/methods/instance/

class mastodon.return_types.**Nodeinfo**(***kwargs*)

The instances standardized NodeInfo data.

Example:

```
# Returns a Nodeinfo object
mastodon.instance_nodeinfo()
```

See also (Mastodon API documentation): https://github.com/jhass/nodeinfo

class mastodon.return_types.**NodeinfoSoftware**(***kwargs*)

NodeInfo software-related information.

Example:

```
# Returns a NodeinfoSoftware object
mastodon.instance_nodeinfo().software
```

See also (Mastodon API documentation): https://github.com/jhass/nodeinfo

class mastodon.return_types.**NodeinfoServices**(***kwargs*)

Nodeinfo services-related information.

Example:

```
# Returns a NodeinfoServices object
mastodon.instance_nodeinfo().services
```

See also (Mastodon API documentation): https://github.com/jhass/nodeinfo

class mastodon.return_types.**NodeinfoUsage**(***kwargs*)

Nodeinfo usage-related information.

Example:

```
# Returns a NodeinfoUsage object
mastodon.instance_nodeinfo().usage
```

See also (Mastodon API documentation): https://github.com/jhass/nodeinfo

class mastodon.return_types.**NodeinfoUsageUsers**(***kwargs*)

Nodeinfo user count statistics.

Example:

```
# Returns a NodeinfoUsageUsers object
mastodon.instance_nodeinfo().usage.users
```

See also (Mastodon API documentation): https://github.com/jhass/nodeinfo

class mastodon.return_types.**NodeinfoMetadata**(***kwargs*)

Nodeinfo extra metadata. Entirely freeform, be careful about consuming it programatically. Survey of real world usage: https://codeberg.org/thefederationinfo/nodeinfo_metadata_survey.

Example:

```
# Returns a NodeinfoMetadata object
mastodon.instance_nodeinfo().metadata
```

See also (Mastodon API documentation): https://github.com/jhass/nodeinfo

class mastodon.return_types.**Activity**(***kwargs*)

Information about recent activity on an instance.

Example:

```
# Returns a Activity object
mastodon.instance_activity()[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/methods/instance/#activity

class mastodon.return_types.**Report**(***kwargs*)

Information about a report that has been filed against a user. Currently largely pointless, as updated reports cannot be fetched.

Example:

```
# Returns a Report object
mastodon.report(<account id>)
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Report/

class mastodon.return_types.**AdminReport**(***kwargs*)

Information about a report that has been filed against a user.

Example:

```
# Returns a AdminReport object
mastodon.admin_reports()[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Admin_Report/

class mastodon.return_types.**WebPushSubscription**(***kwargs*)

Information about the logged-in users web push subscription for the authenticated application.

Example:

```
# Returns a WebPushSubscription object
mastodon.push_subscription()
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/WebPushSubscription/

class mastodon.return_types.**WebPushSubscriptionAlerts**(***kwargs*)

Information about alerts as part of a push subscription.

Example:

```
# Returns a WebPushSubscriptionAlerts object
mastodon.push_subscription().alerts
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/WebPushSubscription/

class mastodon.return_types.**PushNotification**(***kwargs*)

A single Mastodon push notification received via WebPush, after decryption.

Example:

```
# Returns a PushNotification object
mastodon.push_subscription_decrypt_push(...)
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/WebPushSubscription/

class mastodon.return_types.**Preferences**(***kwargs*)

The logged in users preferences.

Example:

```
# Returns a Preferences object
mastodon.preferences()
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Preferences/

class mastodon.return_types.**FeaturedTag**(***kwargs*)

A tag featured on a users profile.

Example:

```
# Returns a FeaturedTag object
mastodon.featured_tags()[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/FeaturedTag/

class mastodon.return_types.**Marker**(***kwargs*)

A read marker indicating where the logged in user has left off reading a given timeline.

Example:

```
# Returns a Marker object
mastodon.markers_get()["home"]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Marker/

class mastodon.return_types.**Announcement**(***kwargs*)

An announcement sent by the instances staff.

Example:

```
# Returns a Announcement object
mastodon.announcements()[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Announcement/

class mastodon.return_types.**Reaction**(***kwargs*)

A reaction to an announcement.

Example:

```
# Returns a Reaction object
mastodon.announcements()[0].reactions[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Reaction/

class mastodon.return_types.**StreamReaction**(***kwargs*)

A reaction to an announcement.

Example:

```
# Returns a StreamReaction object
# Only available via the streaming API
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/methods/streaming/

class mastodon.return_types.**FamiliarFollowers**(***kwargs*)

A follower of a given account that is also followed by the logged-in user.

Example:

```
# Returns a FamiliarFollowers object
mastodon.account_familiar_followers(<account id>)[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/FamiliarFollowers/

class mastodon.return_types.**AdminAccount**(***kwargs*)

Admin variant of the Account entity, with some additional information.

Example:

```
# Returns a AdminAccount object
mastodon.admin_account(<account id>)
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Admin_Account/

class mastodon.return_types.**AdminIp**(***kwargs*)

An IP address used by some user or other instance, visible as part of some admin APIs.

Example:

```
# Returns a AdminIp object
mastodon.admin_account(<account id>).ips[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Admin_Ip/

class mastodon.return_types.**AdminMeasure**(***kwargs*)

A measurement, such as the number of active users, as returned by the admin reporting API.

Example:

```
# Returns a AdminMeasure object
mastodon.admin_measures(datetime.now() - timedelta(hours=24*5), datetime.now(),␣
↪interactions=True)[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Admin_Measure/

class mastodon.return_types.**AdminMeasureData**(***kwargs*)

A single row of data for an admin reporting api measurement.

Example:

```
# Returns a AdminMeasureData object
mastodon.admin_measures(datetime.now() - timedelta(hours=24*5), datetime.now(),␣
↪active_users=True)[0].data[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Admin_Measure/

class mastodon.return_types.**AdminDimension**(***kwargs*)

A qualitative measurement about the server, as returned by the admin reporting api.

Example:

```
# Returns a AdminDimension object
mastodon.admin_dimensions(datetime.now() - timedelta(hours=24*5), datetime.now(),␣
↪languages=True)[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Admin_Dimension/

class mastodon.return_types.**AdminDimensionData**(***kwargs*)

A single row of data for qualitative measurements about the server, as returned by the admin reporting api.

Example:

```
# Returns a AdminDimensionData object
mastodon.admin_dimensions(datetime.now() - timedelta(hours=24*5), datetime.now(),␣
↪languages=True)[0].data[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Admin_Dimension/

class mastodon.return_types.**AdminRetention**(***kwargs*)

User retention data for a given cohort, as returned by the admin reporting api.

Example:

```
# Returns a AdminRetention object
mastodon.admin_retention(datetime.now() - timedelta(hours=24*5), datetime.now())[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Admin_Cohort/

class mastodon.return_types.**AdminCohort**(***kwargs*)

A single data point regarding user retention for a given cohort, as returned by the admin reporting api.

Example:

```
# Returns a AdminCohort object
mastodon.admin_retention(datetime.now() - timedelta(hours=24*5), datetime.now())[0].
↪data[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Admin_Cohort/

class mastodon.return_types.**AdminDomainBlock**(***kwargs*)

A domain block, as returned by the admin API.

Example:

```
# Returns a AdminDomainBlock object
mastodon.admin_domain_blocks()[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Admin_DomainBlock/

class mastodon.return_types.**AdminCanonicalEmailBlock**(***kwargs*)

An e-mail block that has been set up to prevent certain e-mails to be used when signing up, via hash matching.

Example:

```
# Returns a AdminCanonicalEmailBlock object
mastodon.admin_create_canonical_email_block(email=<some email>)
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Admin_CanonicalEmailBlock

class mastodon.return_types.**AdminDomainAllow**(***kwargs*)

The opposite of a domain block, specifically allowing a domain to federate when the instance is in allowlist mode.

Example:

```
# Returns a AdminDomainAllow object
mastodon.admin_domain_allows()[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Admin_DomainAllow

class mastodon.return_types.**AdminEmailDomainBlock**(***kwargs*)

A block that has been set up to prevent e-mails from certain domains to be used when signing up.

Example:

```
# Returns a AdminEmailDomainBlock object
mastodo.admin_email_domain_blocks()[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Admin_EmailDomainBlock

class mastodon.return_types.**AdminEmailDomainBlockHistory**(***kwargs*)

Historic data about attempted signups using e-mails from a given domain.

Example:

```
# Returns a AdminEmailDomainBlockHistory object
mastodo.admin_email_domain_blocks()[0].history[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Admin_EmailDomainBlock

class mastodon.return_types.**AdminIpBlock**(***kwargs*)

An admin IP block, to prevent certain IP addresses or address ranges from accessing the instance.

Example:

```
# Returns a AdminIpBlock object
mastodon.admin_ip_blocks()[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Admin_IpBlock

class mastodon.return_types.**DomainBlock**(***kwargs*)

A domain block that has been implemented by instance staff, limiting the way posts from the blocked instance are handled.

Example:

```
# Returns a DomainBlock object
mastodon.instance_domain_blocks()[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/DomainBlock

class mastodon.return_types.**ExtendedDescription**(***kwargs*)

An extended instance description that can contain HTML.

Example:

```
# Returns a ExtendedDescription object
mastodon.instance_extended_description()
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/ExtendedDescription

class mastodon.return_types.**FilterKeyword**(***kwargs*)

A keyword that is being matched as part of a filter.

Example:

```
# Returns a FilterKeyword object
mastodon.filters_v2()[0].keywords[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/FilterKeyword

class mastodon.return_types.**FilterStatus**(***kwargs*)

A single status that is being matched as part of a filter.

Example:

```
# Returns a FilterStatus object
mastodon.filter_statuses_v2(mastodon.filters_v2()[0])[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/FilterStatus

class mastodon.return_types.**StatusSource**(***kwargs*)

The source data of a status, useful when editing a status.

Example:

```
# Returns a StatusSource object
mastodon.status_source(<status id>)
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/StatusSource

class mastodon.return_types.**Suggestion**(***kwargs*)

A follow suggestion.

Example:

```
# Returns a Suggestion object
mastodon.suggestions_v2()[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Suggestion

class mastodon.return_types.**Translation**(***kwargs*)

A translation of a status.

Example:

```
# Returns a Translation object
mastodon.status_translate(<status_id>, 'de')
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Translation

class mastodon.return_types.**AccountCreationError**(***kwargs*)

An error response returned when creating an account fails.

Example:

```
# Returns a AccountCreationError object
mastodon.create_account('halcy', 'secret', 'invalid email lol', True, return_
↪detailed_error=True)[1]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/methods/accounts/#create

class mastodon.return_types.**AccountCreationErrorDetails**(***kwargs*)

An object containing detailed errors for different fields in the account creation attempt.

Example:

```
# Returns a AccountCreationErrorDetails object
mastodon.create_account('halcy', 'secret', 'invalid email lol', False, return_
↪detailed_error=True)[1].details
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/methods/accounts/#create

class mastodon.return_types.**AccountCreationErrorDetailsField**(***kwargs*)

An object giving details about what specifically is wrong with a given field in an account registration attempt.

Example:

```
# Returns a AccountCreationErrorDetailsField object
mastodon.create_account('halcy', 'secret', 'invalid email lol', True, return_
↪detailed_error=True)[1].details.email[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/methods/accounts/#create

class mastodon.return_types.**NotificationPolicy**(***kwargs*)

Represents the notification filtering policy of the user.

Example:

```
# Returns a NotificationPolicy object
mastodon.notifications_policy()
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/NotificationPolicy

class mastodon.return_types.**NotificationPolicySummary**(***kwargs*)

A summary of the filtered notifications.

Example:

```
# Returns a NotificationPolicySummary object
mastodon.notifications_policy().summary
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/NotificationPolicy

class mastodon.return_types.**RelationshipSeveranceEvent**(***kwargs*)

Summary of a moderation or block event that caused follow relationships to be severed.

Example:

```
# Returns a RelationshipSeveranceEvent object
# There isn't really a good way to get this manually - you get it if a moderation␣
↪takes action.
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/RelationshipSeveranceEvent

class mastodon.return_types.**GroupedNotificationsResults**(***kwargs*)

Container for grouped notifications plus referenced accounts and statuses.

Example:

```
# Returns a GroupedNotificationsResults object
mastodon.grouped_notifications()
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/GroupedNotificationsResults

class mastodon.return_types.**PartialAccountWithAvatar**(***kwargs*)

A stripped-down version of Account, containing only what is necessary to display avatars and a few other fields.

Example:

```
# Returns a PartialAccountWithAvatar object
mastodon.grouped_notifications().partial_accounts[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/GroupedNotificationsResults

class mastodon.return_types.**NotificationGroup**(***kwargs*)

A group of related notifications, plus metadata for pagination.

Example:

```
# Returns a NotificationGroup object
mastodon.grouped_notifications().notification_groups[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/GroupedNotificationsResults

class mastodon.return_types.**AccountWarning**(***kwargs*)

Moderation warning against a particular account.

Example:

```
# Returns a AccountWarning object
# There isn't really a good way to get this manually - you get it if a moderator␣
↪takes action.
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/AccountWarning

class mastodon.return_types.**UnreadNotificationsCount**(***kwargs*)

Rhe (capped) number of unread notifications for the current user.

Example:

```
# Returns a UnreadNotificationsCount object
mastodon.notifications_unread_count()
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/methods/notifications/#unread-count

class mastodon.return_types.**Appeal**(***kwargs*)

Appeal against a moderation action.

Example:

```
# Returns a Appeal object
# There isn't really a good way to get this manually - you get it if a moderator␣
↪takes action.
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Appeal/

class mastodon.return_types.**NotificationRequest**(***kwargs*)

Represents a group of filtered notifications from a specific user.

Example:

```
# Returns a NotificationRequest object
mastodon.notification_requests()[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/NotificationRequest

class mastodon.return_types.**SupportedLocale**(***kwargs*)

A locale supported by the instance.

Example:

```
# Returns a SupportedLocale object
mastodon.instance_languages()[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Instance

class mastodon.return_types.**Filter**(***kwargs*)

Information about a keyword / status filter.

THIS ENTITY IS DEPRECATED. IT IS RECOMMENDED THAT YOU DO NOT USE IT.

Example:

```
# Returns a Filter object
mastodon.filters()[0]
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/V1_Filter/

class mastodon.return_types.**Search**(***kwargs*)

A search result, with accounts, hashtags and statuses.

THIS ENTITY IS DEPRECATED. IT IS RECOMMENDED THAT YOU DO NOT USE IT.

Example:

```
# Returns a Search object
mastodon.search_v1("<search query>")
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/Search/

class mastodon.return_types.**IdentityProof**(***kwargs*)

A cryptographic proof-of-identity. Deprecated since 3.5.0.

THIS ENTITY IS DEPRECATED. IT IS RECOMMENDED THAT YOU DO NOT USE IT.

Example:

```
# Returns a IdentityProof object
# Deprecated since 3.5.0 and eventually removed, there is no way to get this on␣
↪current versions of Mastodon.
```

See also (Mastodon API documentation): https://docs.joinmastodon.org/entities/IdentityProof

static Mastodon.**create_app**(*client_name*, *scopes: List[str] = ['read', 'write', 'follow', 'push']*, *redirect_uris: str | List[str] | None = None*, *website: str | None = None*, *to_file: str | PurePath | None = None*, *api_base_url: str | None = None*, *request_timeout: float = 300*, *session: Session | None = None*, *user_agent: str = 'mastodonpy'*) → Tuple[str, str]

Create a new app with given *client_name* and *scopes* (The basic scopes are "read", "write", "follow" and "push" - more granular scopes are available, please refer to Mastodon documentation for which) on the instance given by *api_base_url*. If you pass scopes, you must pass the same set of scopes to *log_in()* and *auth_request_url()*, otherwise, your auth request will fail.

Specify *redirect_uris* if you want users to be redirected to a certain page after authenticating in an OAuth flow. You can specify multiple URLs by passing a list. Note that if you wish to use OAuth authentication with redirects, the redirect URI must be one of the URLs specified here.

Specify *to_file* to persist your app's info to a file so you can use it in the constructor. Specify *website* to give a website for your app.

Specify *session* with a requests.Session for it to be used instead of the default. This can be used to, amongst other things, adjust proxy or SSL certificate settings.

Specify *user_agent* if you want to use a specific name as *User-Agent* header, otherwise "mastodonpy" will be used.

Presently, app registration is open by default, but this is not guaranteed to be the case for all Mastodon instances in the future.

Returns *client_id* and *client_secret*, both as strings.

Mastodon.**app_verify_credentials**() → *Application*

Fetch information about the current application.

Added: Mastodon v2.0.0, last changed: Mastodon v2.7.2 (parameters), Mastodon v4.3.0 (return value)

Mastodon.**__init__**(*client_id: str | PurePath | None = None*, *client_secret: str | None = None*, *access_token: str | PurePath | None = None*, *api_base_url: str | None = None*, *debug_requests: bool = False*, *ratelimit_method: str = 'wait'*, *ratelimit_pacefactor: float = 1.1*, *request_timeout: float = 300*, *mastodon_version: str | None = None*, *version_check_mode: str = 'none'*, *session: Session | None = None*, *feature_set: str = 'mainline'*, *user_agent: str = 'mastodonpy'*, *lang: str | None = None*)

Create a new API wrapper instance based on the given *client_secret* and *client_id* on the instance given by *api_base_url*. If you give a *client_id* and it is not a file, you must also give a secret. If you specify an *access_token* then you don't need to specify a *client_id*. It is allowed to specify neither - in this case, you will be restricted to only using endpoints that do not require authentication. If a file is given as *client_id*, client ID, secret and base url are read from that file.

You can also specify an *access_token*, directly or as a file (as written by *log_in()*). If a file is given, Mastodon.py also tries to load the base URL from this file, if present. A client id and secret are not required in this case.

Mastodon.py can try to respect rate limits in several ways, controlled by *ratelimit_method*. "throw" makes functions throw a *MastodonRatelimitError* when the rate limit is hit. "wait" mode will, once the limit is hit, wait and retry the request as soon as the rate limit resets, until it succeeds. "pace" works like throw, but tries to wait in between calls so that the limit is generally not hit (how hard it tries to avoid hitting the rate limit can be controlled by ratelimit_pacefactor). The default setting is "wait". Note that even in "wait" and "pace" mode, requests can still fail due to network or other problems! Also note that "pace" and "wait" are NOT thread safe.

By default, a timeout of 300 seconds is used for all requests. If you wish to change this, pass the desired timeout (in seconds) as *request_timeout*.

For fine-tuned control over the requests object use *session* with a requests.Session.

The *mastodon_version* parameter can be used to specify the version of Mastodon that Mastodon.py will expect to be installed on the server. The function will throw an error if an unparseable Version is specified. If no version is specified, Mastodon.py will set *mastodon_version* to the detected version.

feature_set can be used to enable behaviour specific to non-mainline Mastodon API implementations. Details are documented in the functions that provide such functionality. Currently supported feature sets are *mainline*, *fedibird* and *pleroma*.

For some Mastodon instances a *User-Agent* header is needed. This can be set by parameter *user_agent*. Starting from Mastodon.py 1.5.2 *create_app()* stores the application name into the client secret file. If *client_id* points to this file, the app name will be used as *User-Agent* header as default. It is possible to modify old secret files and append a client app name to use it as a *User-Agent* name.

lang can be used to change the locale Mastodon will use to generate responses. Valid parameters are all ISO 639-1 (two letter) or for a language that has none, 639-3 (three letter) language codes. This affects some error messages (those related to validation) and trends. You can change the language using set_language().

The version check mode can be set to "none" (now the default behaviour), "changed" or "created". If set to "created", Mastodon.py will throw an error if the version of Mastodon it is connected to is too old to have an endpoint. If it is set to "changed", it will throw an error if the endpoint's behaviour has changed after the version of Mastodon that is connected has been released. If it is set to "none", version checking is disabled. When encountering problems, I would recommend setting this to "created" and/or setting *debug_requests* to True to get a better idea of what is going on.

If no other *User-Agent* is specified, "mastodonpy" will be used.

Mastodon.**log_in**(*username: str | None = None*, *password: str | None = None*, *code: str | None = None*, *redirect_uri: str = 'urn:ietf:wg:oauth:2.0:oob'*, *refresh_token: str | None = None*, *scopes: List[str] = ['read', 'write', 'follow', 'push']*, *to_file: str | PurePath | None = None*, *allow_http: bool = False*) → str

Get the access token for a user, either via OAuth code flow, or (deprecated) password flow.

Will throw a *MastodonIllegalArgumentError* if the OAuth flow data is incorrect, and *MastodonAPIError* if all of the requested scopes were not granted.

For OAuth2, obtain a code via having your user go to the URL returned by *auth_request_url()* and pass it as the code parameter. In this case, make sure to also pass the same redirect_uri parameter as you used when generating the auth request URL, as well as the same set of scopes, or else your auth request will fail. If passing *code `you should not pass `username* or *password*.

When using the password flow, the username is the email address used to log in into Mastodon. **Note that Mastodon has removed this flow starting with 4.4.0, so it is unfortunately not possible to log in in this way anymore. Please use either the code flow, or generate a token from the web UI.**

Can persist access token to file *to_file*, to be used in the constructor. Pass *allow_http* to allow HTTP URLs for the OAuth server, which is recommended only for testing.

Returns the access token as a string.

Mastodon.**auth_request_url**(*client_id: str | PurePath | None = None*, *redirect_uris: str = 'urn:ietf:wg:oauth:2.0:oob'*, *scopes: List[str] = ['read', 'write', 'follow', 'push']*, *force_login: bool = False*, *state: str | None = None*, *lang: str | None = None*, *skip_server_info=False*, *allow_http: bool = False*) → str

Returns the URL that a client needs to request an OAuth grant from the server.

To log in with OAuth, send your user to this URL. The user will then log in and get a code which you can pass to *log_in()*.

scopes are as in *log_in()*, redirect_uris is where the user should be redirected to after authentication. Note that *redirect_uris* must be one of the URLs given during app registration, and that despite the plural-like name, you only get to use one here. When using urn:ietf:wg:oauth:2.0:oob, the code is simply displayed, otherwise it is added to the given URL as the "code" request parameter. Note that if you pass scopes, you MUST pass the same set of scopes to *log_in()* and *create_app()* <*create_app()*>, otherwise, your auth request will fail.

Pass force_login if you want the user to always log in even when already logged into web Mastodon (i.e. when registering multiple different accounts in an app).

state is the oauth *state* parameter to pass to the server. It is strongly suggested to use a random, nonguessable value (i.e. nothing meaningful and no incrementing ID) to preserve security guarantees. It can be left out for non-web login flows.

Pass an ISO 639-1 (two letter) or, for languages that do not have one, 639-3 (three letter) language code as *lang* to control the display language for the oauth form.

Pass *skip_server_info* to skip retrieving the OAuth authorization server info, in case you want to avoid the extra network request and are confident that the oauth server is at the default location.

Mastodon.**set_language**(*lang: str*)

Set the locale Mastodon will use to generate responses. Valid parameters are all ISO 639-1 (two letter) or, for languages that do not have one, 639-3 (three letter) language codes. This affects some error messages (those related to validation) and trends.

Mastodon.**revoke_access_token**(*allow_http: bool = False*)

Revoke the oauth token the user is currently authenticated with, effectively removing the apps access and requiring the user to log in again.

Mastodon.**create_account**(*username: str*, *password: str*, *email: str*, *agreement: bool = False*, *reason: str | None = None*, *locale: str = 'en'*, *scopes: List[str] = ['read', 'write', 'follow', 'push']*, *to_file: str | None = None*, *return_detailed_error: bool = False*, *date_of_birth: datetime | None = None*) → str | None | Tuple[str | None, *AccountCreationError*]

Creates a new user account with the given username, password and email. "agreement" must be set to true (after showing the user the instance's user agreement and having them agree to it), "locale" specifies the language for the confirmation email as an ISO 639-1 (two letter) or, if a language does not have one, 639-3 (three letter) language code. *reason* can be used to specify why a user would like to join if approved-registrations mode is on. date_of_birth can be used to specify the date of birth of the user, which is required if the server has a minimum age requirement set.

Does not require an access token, but does require a client grant.

By default, this method is rate-limited by IP to 5 requests per 30 minutes.

Returns an access token (just like log_in), which it can also persist to to_file, and sets it internally so that the user is now logged in. Note that this token can only be used after the user has confirmed their email.

By default, the function will throw if the account could not be created. Alternately, when *return_detailed_error* is passed, Mastodon.py will return the detailed error response that the API provides (Starting from version 3.4.0 - not checked here) as an dict with error details as the second return value and the token returned as *None* in case of error. The dict will contain a text *error* values as well as a *details* value which is a dict with one optional key for each potential field (*username*, *password*, *email* and *agreement*), each if present containing a dict with an *error* category and free text *description*. Valid error categories are:

- ERR_BLOCKED - When e-mail provider is not allowed
- ERR_UNREACHABLE - When e-mail address does not resolve to any IP via DNS (MX, A, AAAA)
- ERR_TAKEN - When username or e-mail are already taken

- ERR_RESERVED - When a username is reserved, e.g. "webmaster" or "admin"
- ERR_ACCEPTED - When agreement has not been accepted
- ERR_BLANK - When a required attribute is blank
- ERR_INVALID - When an attribute is malformed, e.g. wrong characters or invalid e-mail address
- ERR_TOO_LONG - When an attribute is over the character limit
- ERR_TOO_SHORT - When an attribute is under the character requirement
- ERR_INCLUSION - When an attribute is not one of the allowed values, e.g. unsupported locale

Added: Mastodon v2.7.0, last changed: Mastodon v2.7.0

Mastodon.**email_resend_confirmation**()

Requests a re-send of the users confirmation mail for an unconfirmed logged in user.

Only available to the app that the user originally signed up with.

Added: Mastodon v3.4.0, last changed: Mastodon v3.4.0

Mastodon.**preferences**() → *Preferences*

Fetch the user's preferences, which can be used to set some default options. As of 2.8.0, apps can only fetch, not update preferences.

Added: Mastodon v2.8.0, last changed: Mastodon v2.8.0 (parameters), Mastodon v2.8.0 (return value)

Mastodon.**status**(*id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Status*

Fetch information about a single toot.

Does not require authentication for publicly visible statuses.

Added: Mastodon v1.0.0, last changed: Mastodon v2.0.0 (parameters), Mastodon v4.4.0 (return value)

Mastodon.**status_context**(*id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Context*

Fetch information about ancestors and descendants of a toot.

Does not require authentication for publicly visible statuses.

Added: Mastodon v1.0.0, last changed: Mastodon v1.0.0 (parameters), Mastodon v0.6.0 (return value)

Mastodon.**status_reblogged_by**(*id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *NonPaginatableList*[*Account*]

Fetch a list of users that have reblogged a status.

Does not require authentication for publicly visible statuses.

Interesting caveat: If you self-reblog a status with private visibility, this endpoint will not return your account as having reblogged it.

Added: Mastodon v1.0.0, last changed: Mastodon v2.1.0

Mastodon.**status_favourited_by**(*id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *NonPaginatableList*[*Account*]

Fetch a list of users that have favourited a status.

Does not require authentication for publicly visible statuses.

Added: Mastodon v1.0.0, last changed: Mastodon v2.1.0

Mastodon.**status_card**(*id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *PreviewCard*

Fetch a card associated with a status. A card describes an object (such as an external video or link) embedded into a status.

Does not require authentication for publicly visible statuses.

This function is deprecated as of 3.0.0 and the endpoint does not exist anymore - you should just use the "card" field of the status instead. Mastodon.py will try to mimic the old behaviour, but this is somewhat inefficient and not guaranteed to be the case forever.

Added: Mastodon v1.0.0, last changed: Mastodon v3.0.0 (parameters), Mastodon v4.3.0 (return value)

Mastodon.**status_history**(*id:* StatusEdit | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *NonPaginatableList*[*StatusEdit*]

Returns the edit history of a status as a list of StatusEdit objects, starting from the original form. Note that this means that a status that has been edited once will have *two* entries in this list, a status that has been edited twice will have three, and so on.

Added: Mastodon v3.5.0, last changed: Mastodon v3.5.0

Mastodon.**status_source**(*id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *StatusSource*

Returns the source of a status for editing.

Return value is a dictionary containing exactly the parameters you could pass to *status_update()* to change nothing about the status, except *status* is *text* instead.

Mastodon.**statuses**(*ids: List[*Status | *str* | *int* | MaybeSnowflakeIdType | *datetime]*) → List[*Status*]

Fetch information from multiple statuses by a list of status *id*.

Does not require authentication for publicly visible accounts.

Added: Mastodon v4.3.0, last changed: Mastodon v4.3.0

Mastodon.**favourites**(*max_id: str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *min_id: str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *since_id: str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *limit: int* | *None* = *None*) → *PaginatableList*[*Status*]

Fetch the logged-in user's favourited statuses.

This endpoint uses internal ids for pagination, passing status ids to *max_id*, *min_id*, or *since_id* will not work.

Added: Mastodon v1.0.0, last changed: Mastodon v2.6.0

Mastodon.**bookmarks**(*max_id: str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *min_id: str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *since_id: str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *limit: int* | *None* = *None*) → *PaginatableList*[*Status*]

Get a list of statuses bookmarked by the logged-in user.

This endpoint uses internal ids for pagination, passing status ids to *max_id*, *min_id*, or *since_id* will not work.

Added: Mastodon v3.1.0, last changed: Mastodon v3.1.0

Mastodon.**status_post**(*status: str*, *in_reply_to_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *media_ids: List[*MediaAttachment | *str* | *int* | MaybeSnowflakeIdType | *datetime]* | *None* = *None*, *sensitive: bool* = *False*, *visibility: str* | *None* = *None*, *spoiler_text: str* | *None* = *None*, *language: str* | *None* = *None*, *idempotency_key: str* | *None* = *None*, *content_type: str* | *None* = *None*, *scheduled_at: datetime* | *None* = *None*, *poll:* Poll | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *quote_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *strict_content_type: bool* = *False*) → *Status* | *ScheduledStatus*

Post a status. Can optionally be in reply to another status and contain media.

media_ids should be a list. (If it's not, the function will turn it into one.) It can contain up to four pieces of media (uploaded via *media_post()*). *media_ids* can also be the objects returned by *media_post()* - they are unpacked automatically.

The *sensitive* boolean decides whether or not media attached to the post should be marked as sensitive, which hides it by default on the Mastodon web front-end.

The *visibility* parameter is a string value and accepts any of:

- `'direct'` - post will be visible only to **mentioned users**, known in Mastodon's UI as "Mentioned users only"
- `'private'` - post will be visible only to **followers**, known in Mastodon's UI as "Followers only"
- `'unlisted'` - post will be public but **will not appear** on the public timelines
- `'public'` - post will be public and **will appear** on public timelines

If not passed in, *visibility* defaults to match the current account's default-privacy setting (starting with Mastodon version 1.6) or its locked setting - `'private'` if the account is locked, `'public'` otherwise (for Mastodon versions lower than 1.6).

The *spoiler_text* parameter is a string to be shown as a warning before the text of the status. If no text is passed in, no warning will be displayed.

Specify *language* to override automatic language detection. The parameter accepts all valid ISO 639-1 (2-letter) or for languages where that do not have one, 639-3 (three letter) language codes.

You can set *idempotency_key* to a value to uniquely identify an attempt at posting a status. Even if you call this function more than once, if you call it with the same *idempotency_key*, only one status will be created.

Pass a datetime as *scheduled_at* to schedule the toot for a specific time (the time must be at least 5 minutes into the future). If this is passed, status_post returns a *ScheduledStatus* instead.

Pass *poll* to attach a poll to the status. An appropriate object can be constructed using *make_poll()* . Note that as of Mastodon version 2.8.2, you can only have either media or a poll attached, not both at the same time.

You can use *get_status_length()* to count how many characters a status you want to post would take up in terms of Mastodons character limit. The limits can be retrieved from the instance information (*instance_v2()*).

Specific to "pleroma" feature set:: Specify *content_type* to set the content type of your post on Pleroma. It accepts 'text/plain' (default), 'text/markdown', 'text/html' and 'text/bbcode'. This parameter is not supported on Mastodon servers, but will be safely ignored if set. If you want to throw an error if the content type is not known to work on the server, set *strict_content_type* to True.

Specific to "fedibird" feature set:: The *quote_id* parameter is a non-standard extension that specifies the id of a quoted status.

Returns the new status.

Added: Mastodon v1.0.0, last changed: Mastodon v2.8.0

`Mastodon.status_reply`(*to_status:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *status: str*, *media_ids:* *List[*MediaAttachment | *str* | *int* | MaybeSnowflakeIdType | *datetime]* | *None = None*, *sensitive: bool = False*, *visibility: str* | *None = None*, *spoiler_text: str* | *None = None*, *language: str* | *None = None*, *idempotency_key: str* | *None = None*, *content_type: str* | *None = None*, *scheduled_at: datetime* | *None = None*, *poll:* Poll | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *quote_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *untag: bool = False*, *strict_content_type: bool = False*) → *Status*

Helper function - acts like status_post, but prepends the name of all the users that are being replied to the status text and retains CW and visibility if not explicitly overridden.

Note that *to_status* must be a *Status* and not just an ID.

Set *untag* to True if you want the reply to only go to the user you are replying to, removing every other mentioned user from the conversation.

Added: Mastodon v1.0.0, last changed: Mastodon v2.8.0 (parameters), Mastodon v4.4.0 (return value)

`Mastodon.`**`toot`**(*status: str*) → *Status*

Synonym for *status_post()* that only takes the status text as input.

Usage in production code is not recommended.

Added: Mastodon v1.0.0, last changed: Mastodon v2.8.0 (parameters), Mastodon v4.4.0 (return value)

`Mastodon.`**`make_poll`**(*options: List[str]*, *expires_in: int*, *multiple: bool = False*, *hide_totals: bool = False*) → *Poll*

Generate a poll object that can be passed as the *poll* option when posting a status.

options is an array of strings with the poll options (Maximum, by default: 4 - see the instance configuration for the actual value on any given instance, if stated). *expires_in* is the time in seconds for which the poll should be open. Set multiple to True to allow people to choose more than one answer. Set hide_totals to True to hide the results of the poll until it has expired.

Added: Mastodon v2.8.0, last changed: Mastodon v2.8.0 (parameters), Mastodon v2.8.0 (return value)

`Mastodon.`**`status_reblog`**(*id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *visibility: str | None = None*) → *Status*

Reblog / boost a status.

The visibility parameter functions the same as in *status_post()* and allows you to reduce the visibility of a reblogged status.

Returns a new Status that wraps around the reblogged status.

Added: Mastodon v1.0.0, last changed: Mastodon v2.0.0 (parameters), Mastodon v4.4.0 (return value)

`Mastodon.`**`status_unreblog`**(*id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Status*

Un-reblog a status.

Returns the status that used to be reblogged.

Added: Mastodon v1.0.0, last changed: Mastodon v2.0.0 (parameters), Mastodon v4.4.0 (return value)

`Mastodon.`**`status_favourite`**(*id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Status*

Favourite a status.

Returns the favourited status.

Added: Mastodon v1.0.0, last changed: Mastodon v2.0.0 (parameters), Mastodon v4.4.0 (return value)

`Mastodon.`**`status_unfavourite`**(*id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Status*

Un-favourite a status.

Returns the un-favourited status.

Added: Mastodon v1.0.0, last changed: Mastodon v2.0.0 (parameters), Mastodon v4.4.0 (return value)

Mastodon.**status_mute**(*id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Status*

Mute notifications for a status.

Returns the now muted status

Added: Mastodon v1.4.0, last changed: Mastodon v2.0.0 (parameters), Mastodon v4.4.0 (return value)

Mastodon.**status_unmute**(*id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Status*

Unmute notifications for a status.

Returns the status that used to be muted.

Added: Mastodon v1.4.0, last changed: Mastodon v2.0.0 (parameters), Mastodon v4.4.0 (return value)

Mastodon.**status_bookmark**(*id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Status*

Bookmark a status as the logged-in user.

Returns the now bookmarked status

Added: Mastodon v3.1.0, last changed: Mastodon v3.1.0 (parameters), Mastodon v4.4.0 (return value)

Mastodon.**status_unbookmark**(*id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Status*

Unbookmark a bookmarked status for the logged-in user.

Returns the status that used to be bookmarked.

Added: Mastodon v3.1.0, last changed: Mastodon v3.1.0 (parameters), Mastodon v4.4.0 (return value)

Mastodon.**status_delete**(*id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *delete_media: bool = None*) → *Status*

Delete a status

Returns the now-deleted status, with an added "text" attribute that contains the text that was used to compose this status (this can be used to power "delete and redraft" functionality) as well as either poll or media_attachments set in the same way. Note that when reattaching media, you have to wait up to several seconds for the media to be un-attached from the original status - that operation is not synchronous with the delete.

Pass *delete_media=True* to delete the media attachments of the status immediately, instead of just scheduling them for deletion as part of the next media cleanup. If you set this, you will not be able to reuse them in a new status (so if you're delete-redrafting, you should not set this).

Added: Mastodon v1.0.0, last changed: Mastodon v1.0.0 (parameters), Mastodon v4.4.0 (return value)

Mastodon.**status_update**(*id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *status: str*, *spoiler_text: str | None = None*, *sensitive: bool | None = None*, *media_ids: List[*MediaAttachment | *str | int* | MaybeSnowflakeIdType | *datetime] | None = None*, *poll:* Poll | *str | int |* MaybeSnowflakeIdType | *datetime | None = None*, *media_attributes: List[Dict[str, Any]] | None = None*) → *Status*

Edit a status. The meanings of the fields are largely the same as in *status_post()*, though not every field can be edited. The *status* parameter is mandatory.

Note that editing a poll will reset the votes.

To edit media metadata, generate a list of dictionaries with the following keys:

Added: Mastodon v3.5.0, last changed: Mastodon v4.1.0 (parameters), Mastodon v4.4.0 (return value)

Mastodon.**generate_media_edit_attributes**(*id:* MediaAttachment | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *description: str | None = None*, *focus: Tuple[float, float] | None = None*, *thumbnail: str | PurePath | IO[bytes] | None = None*, *thumb_mimetype: str | None = None*) → Dict[str, Any]

Helper function to generate a single media edit attribute dictionary.

Parameters: - *id* (str): The ID of the media attachment (mandatory). - *description* (Optional[str]): A new description for the media. - *focus* (Optional[Tuple[float, float]]): The focal point of the media. - *thumbnail* (Optional[PathOrFile]): The thumbnail to be used.

Mastodon.**scheduled_statuses**(*max_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *min_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *since_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *limit: int* | *None* = *None*) → *PaginatableList*[*ScheduledStatus*]

Fetch a list of scheduled statuses

Added: Mastodon v2.7.0, last changed: Mastodon v2.7.0

Mastodon.**scheduled_status**(*id:* ScheduledStatus | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *ScheduledStatus*

Fetch information about the scheduled status with the given id.

Added: Mastodon v2.7.0, last changed: Mastodon v2.7.0 (parameters), Mastodon v2.7.0 (return value)

Mastodon.**scheduled_status_update**(*id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *scheduled_at: datetime*) → *ScheduledStatus*

Update the scheduled time of a scheduled status.

New time must be at least 5 minutes into the future.

Returned object reflects the updates to the scheduled status.

Added: Mastodon v2.7.0, last changed: Mastodon v2.7.0 (parameters), Mastodon v2.7.0 (return value)

Mastodon.**scheduled_status_delete**(*id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime*)

Deletes a scheduled status.

Added: Mastodon v2.7.0, last changed: Mastodon v2.7.0

Mastodon.**media_post**(*media_file: str* | *PurePath* | *IO[bytes]*, *mime_type: str* | *None* = *None*, *description: str* | *None* = *None*, *focus: Tuple[float, float]* | *None* = *None*, *file_name: str* | *None* = *None*, *thumbnail: str* | *PurePath* | *IO[bytes]* | *None* = *None*, *thumbnail_mime_type: str* | *None* = *None*, *synchronous: bool* = *False*) → *MediaAttachment*

Post an image, video or audio file. *media_file* can either be data or a file name. If data is passed directly, the mime type has to be specified manually, otherwise, it is determined from the file name. *focus* should be a tuple of floats between -1 and 1, giving the x and y coordinates of the images focus point for cropping (with the origin being the images center).

Throws a *MastodonIllegalArgumentError* if the mime type of the passed data or file can not be determined properly.

file_name can be specified to upload a file with the given name, which is ignored by Mastodon, but some other Fediverse server software will display it. If no name is specified, a random name will be generated. The filename of a file specified in media_file will be ignored.

Starting with Mastodon 3.2.0, *thumbnail* can be specified in the same way as *media_file* to upload a custom thumbnail image for audio and video files.

When using the v2 API (post Mastodon version 3.1.4), the *url* in the returned dict will be *null*, since attachments are processed asynchronously. You can fetch an updated dict using *media*. Pass "synchronous" to emulate the old behaviour. Not recommended, inefficient and deprecated, will eat your API quota, you know the deal.

The returned value (or its id) can be used in a status post as the *media_ids* parameter.

Added: Mastodon v1.0.0, last changed: Mastodon v3.2.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**media_update**(*id:* MediaAttachment | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *description: str* | *None = None*, *focus: Tuple[float, float]* | *None = None*, *thumbnail: str* | *PurePath* | *IO[bytes]* | *None = None*, *thumbnail_mime_type=None*) → *MediaAttachment*

Update the metadata of the media file with the given *id*. *description* and *focus* and *thumbnail* are as in *media_post()* .

The returned dict reflects the updates to the media attachment.

Note: This is for updating the metadata of an *unattached* media attachment (i.e. one that has not been used in a status yet). For editing media attachment metadata after posting, see *status_update* and *generate_media_edit_attributes*.

Added: Mastodon v2.3.0, last changed: Mastodon v3.2.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**media**(*id:* MediaAttachment | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *MediaAttachment*

Get the updated JSON for one non-attached / in progress media upload belonging to the logged-in user.

Added: Mastodon v3.1.4, last changed: Mastodon v3.1.4 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**poll**(*id:* Poll | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Poll*

Fetch information about the poll with the given id

Added: Mastodon v2.8.0, last changed: Mastodon v2.8.0 (parameters), Mastodon v2.8.0 (return value)

Mastodon.**poll_vote**(*id:* Poll | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *choices: int* | *List[int]*) → *Poll*

Vote in the given poll.

choices is the index of the choice you wish to register a vote for (i.e. its index in the corresponding polls *options* field. In case of a poll that allows selection of more than one option, a list of indices can be passed.

You can only submit choices for any given poll once in case of single-option polls, or only once per option in case of multi-option polls.

The returned object will reflect the updated votes.

Added: Mastodon v2.8.0, last changed: Mastodon v2.8.0 (parameters), Mastodon v2.8.0 (return value)

Mastodon.**status_translate**(*id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *lang: str* | *None = None*) → *Translation*

Translate the status content into some language.

Raises a MastodonAPIError if the server can't perform the requested translation, for any reason (doesn't support translation, unsupported language pair, etc.).

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**account_verify_credentials**() → *Account*

Fetch logged-in user's account information. Returns the version of the Account object with *source* field.

Added: Mastodon v1.0.0, last changed: Mastodon v2.1.0 (parameters), Mastodon v4.2.0 (return value)

Mastodon.**me**() → *Account*

Get this user's account. Synonym for *account_verify_credentials()*, does exactly the same thing, just exists because *account_verify_credentials()* has a confusing name.

Added: Mastodon v1.0.0, last changed: Mastodon v2.1.0 (parameters), Mastodon v4.2.0 (return value)

Mastodon.**account**(*id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Account*

Fetch account information by user *id*.

Does not require authentication for publicly visible accounts.

Added: Mastodon v1.0.0, last changed: Mastodon v1.0.0 (parameters), Mastodon v4.2.0 (return value)

Mastodon.**account_search**(*q: str*, *limit: int | None = None*, *following: bool = False*, *resolve: bool = False*, *offset: int | None = None*) → *NonPaginatableList*[*Account*]

Fetch matching accounts. Will lookup an account remotely if the search term is in the username@domain format and not yet in the database. Set *following* to True to limit the search to users the logged-in user follows.

Paginated in a weird way ("limit" / "offset"), if you want to fetch all results here please do it yourself for now.

Added: Mastodon v1.0.0, last changed: Mastodon v2.8.0

Mastodon.**account_lookup**(*acct: str*) → *Account*

Look up an account from user@instance form (@instance allowed but not required for local accounts). Will only return accounts that the instance already knows about, and not do any webfinger requests. Use *account_search* if you need to resolve users through webfinger from remote.

Added: Mastodon v3.4.0, last changed: Mastodon v3.4.0 (parameters), Mastodon v4.2.0 (return value)

Mastodon.**accounts**(*ids: List[*Account | *str* | *int* | MaybeSnowflakeIdType | *datetime]*) → List[*Account*]

Fetch information from multiple accounts by a list of user *id*.

Does not require authentication for publicly visible accounts.

Added: Mastodon v4.3.0, last changed: Mastodon v4.3.0

Mastodon.**featured_tags**() → *NonPaginatableList*[*FeaturedTag*]

Return the hashtags the logged-in user has set to be featured on their profile.

Added: Mastodon v3.0.0, last changed: Mastodon v3.0.0

Mastodon.**featured_tag_suggestions**() → *NonPaginatableList*[*FeaturedTag*]

Returns the logged-in user's 10 most commonly-used hashtags.

Added: Mastodon v3.0.0, last changed: Mastodon v3.0.0

Mastodon.**account_featured_tags**(*id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *NonPaginatableList*[*Tag*]

Get an account's featured hashtags.

Added: Mastodon v3.3.0, last changed: Mastodon v3.3.0

Mastodon.**endorsements**() → *NonPaginatableList*[*Account*]

Fetch list of users endorsed by the logged-in user.

Added: Mastodon v2.5.0, last changed: Mastodon v2.5.0

Mastodon.**account_statuses**(*id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *only_media: bool = False*, *pinned: bool = False*, *exclude_replies: bool = False*, *exclude_reblogs: bool = False*, *tagged: str | None = None*, *max_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *min_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *since_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *limit: int | None = None*) → *PaginatableList*[*Status*]

Fetch statuses by user *id*. Same options as *timeline()* are permitted. Returned toots are from the perspective of the logged-in user, i.e. all statuses visible to the logged-in user (including DMs) are included.

If *only_media* is set, return only statuses with media attachments. If *pinned* is set, return only statuses that have been pinned. Note that as of Mastodon 2.1.0, this only works properly for instance-local users. If *exclude_replies* is set, filter out all statuses that are replies. If *exclude_reblogs* is set, filter out all statuses that are reblogs. If *tagged* is set, return only statuses that are tagged with *tagged*. Only a single tag without a '#' is valid.

Does not require authentication for Mastodon versions after 2.7.0 (returns publicly visible statuses in that case), for publicly visible accounts.

Added: Mastodon v1.0.0, last changed: Mastodon v2.8.0

Mastodon.**account_familiar_followers**(*id: List[*Account | *str* | *int* | MaybeSnowflakeIdType | *datetime]* | Account | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *NonPaginatableList*[*FamiliarFollowers*]

Find followers for the account given by id (can be a list) that also follow the logged in account.

Added: Mastodon v3.5.0, last changed: Mastodon v3.5.0

Mastodon.**account_lists**(*id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *NonPaginatableList*[*UserList*]

Get all of the logged-in user's lists which the specified user is a member of.

Added: Mastodon v2.1.0, last changed: Mastodon v2.1.0

Mastodon.**account_update_credentials**(*display_name: str | None = None*, *note: str | None = None*, *avatar: str | PurePath | IO[bytes] | None = None*, *avatar_mime_type: str | None = None*, *header: str | PurePath | IO[bytes] | None = None*, *header_mime_type: str | None = None*, *locked: bool | None = None*, *bot: bool | None = None*, *discoverable: bool | None = None*, *fields: List[Tuple[str, str]] | None = None*, *attribution_domains: List[str] | None = None*) → *Account*

Update the profile for the currently logged-in user.

note is the user's bio.

avatar and 'header' are images. As with media uploads, it is possible to either pass image data and a mime type, or a filename of an image file, for either.

locked specifies whether the user needs to manually approve follow requests.

bot specifies whether the user should be set to a bot.

discoverable specifies whether the user should appear in the user directory.

fields can be a list of up to four name-value pairs (specified as tuples) to appear as semi-structured information in the user's profile.

attribution_domains can be a list of domains that the user wants to allow to attribute content to them.

The returned object reflects the updated account.

Added: Mastodon v1.1.1, last changed: Mastodon v3.1.0 (parameters), Mastodon v4.2.0 (return value)

Mastodon.**account_pin**(*id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Relationship*

Pin / endorse a user.

The returned object reflects the updated relationship with the user.

Deprecated, use *account_endorse* instead.

Added: Mastodon v2.5.0, last changed: Mastodon v2.5.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**account_unpin**(*id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Relationship*

Unpin / un-endorse a user.

The returned object reflects the updated relationship with the user.

Deprecated, use *account_unendorse* instead.

Added: Mastodon v2.5.0, last changed: Mastodon v2.5.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**account_note_set**(*id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *comment: str*) → *Relationship*

Set a note (visible to the logged in user only) for the given account.

The returned object contains the updated note.

nb: To retrieve the current note for an account, use *account_relationships*.

Added: Mastodon v3.2.0, last changed: Mastodon v3.2.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**featured_tag_create**(*name: str*) → *FeaturedTag*

Creates a new featured hashtag displayed on the logged-in user's profile.

The returned object is the newly featured tag.

Obsoleted by *tag_feature* / *tag_unfeature*.

Added: Mastodon v3.0.0, last changed: Mastodon v3.0.0 (parameters), Mastodon v3.3.0 (return value)

Mastodon.**featured_tag_delete**(*id:* FeaturedTag | *str* | *int* | MaybeSnowflakeIdType | *datetime*)

Deletes one of the logged-in user's featured hashtags.

Obsoleted by *tag_feature* / *tag_unfeature*.

Added: Mastodon v3.0.0, last changed: Mastodon v3.0.0

Mastodon.**status_pin**(*id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Status*

Pin a status for the logged-in user.

Returns the now pinned status

Added: Mastodon v2.1.0, last changed: Mastodon v2.1.0 (parameters), Mastodon v4.4.0 (return value)

Mastodon.**status_unpin**(*id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Status*

Unpin a pinned status for the logged-in user.

Returns the status that used to be pinned.

Added: Mastodon v2.1.0, last changed: Mastodon v2.1.0 (parameters), Mastodon v4.4.0 (return value)

Mastodon.**account_delete_avatar**()

Delete the logged-in user's avatar.

Added: Mastodon v4.2.0, last changed: Mastodon v4.2.0

Mastodon.**account_delete_header**()

Delete the logged-in user's header.

Added: Mastodon v4.2.0, last changed: Mastodon v4.2.0

Mastodon.**account_followers**(*id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *max_id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *min_id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *since_id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *limit: int* | *None* = *None*) → *PaginatableList*[*Account*]

Fetch users the given user is followed by.

Added: Mastodon v1.0.0, last changed: Mastodon v2.6.0

Mastodon.**account_following**(*id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *max_id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *min_id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *since_id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *limit: int* | *None* = *None*) → *PaginatableList*[*Account*]

Fetch users the given user is following.

Added: Mastodon v1.0.0, last changed: Mastodon v2.6.0

Mastodon.**account_relationships**(*id: List[*Account | *str* | *int* | MaybeSnowflakeIdType | *datetime]* | Account | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *with_suspended: bool* | *None* = *None*) → *NonPaginatableList*[*Relationship*]

Fetch relationship (following, followed_by, blocking, follow requested) of the logged in user to a given account. *id* can be a list.

Pass *with_suspended* = *True* to include relationships with suspended accounts.

Added: Mastodon v1.0.0, last changed: Mastodon v1.4.0

Mastodon.**follows**(*uri: str*) → *Relationship*

Follow a remote user with username given in username@domain form.

Deprecated - avoid using this. Currently uses a backwards compat implementation that may or may not work properly.

Added: Mastodon v1.0.0, last changed: Mastodon v2.1.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**follow_requests**(*max_id: str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *min_id: str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *since_id: str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *limit: int* | *None* = *None*) → *PaginatableList*[*Account*]

Fetch the logged-in user's incoming follow requests.

Added: Mastodon v1.0.0, last changed: Mastodon v2.6.0

Mastodon.**suggestions**() → *NonPaginatableList*[*Suggestion*] | *NonPaginatableList*[*Account*]

Fetch follow suggestions for the logged-in user.

Will use the v1 endpoint if the server is below 3.4.0, otherwise will use the v2 endpoint and unpack the account dicts.

Mastodon.**account_follow**(*id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *reblogs: bool* = *True*, *notify: bool* = *False*) → *Relationship*

Follow a user.

Set *reblogs* to False to hide boosts by the followed user. Set *notify* to True to get a notification every time the followed user posts.

The returned object reflects the updated relationship with the user.

Added: Mastodon v1.0.0, last changed: Mastodon v3.3.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**account_unfollow**(*id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Relationship*

Unfollow a user.

The returned object reflects the updated relationship with the user.

Added: Mastodon v1.0.0, last changed: Mastodon v1.4.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**follow_request_authorize**(*id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Relationship*

Accept an incoming follow request from the given Account and returns the updated Relationship.

Added: Mastodon v1.0.0, last changed: Mastodon v3.0.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**follow_request_reject**(*id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Relationship*

Reject an incoming follow request from the given Account and returns the updated Relationship.

Added: Mastodon v1.0.0, last changed: Mastodon v3.0.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**suggestion_delete**(*account_id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime*)

Remove the user with the given *account_id* from the follow suggestions.

Added: Mastodon v2.4.3, last changed: Mastodon v2.4.3

Mastodon.**mutes**(*max_id: str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *min_id: str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *since_id: str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *limit: int* | *None* = *None*) → *PaginatableList*[*Account*]

Fetch a list of users muted by the logged-in user.

Added: Mastodon v1.1.0, last changed: Mastodon v2.6.0

Mastodon.**blocks**(*max_id: str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *min_id: str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *since_id: str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *limit: int* | *None* = *None*) → *PaginatableList*[*Account*]

Fetch a list of users blocked by the logged-in user.

Added: Mastodon v1.0.0, last changed: Mastodon v2.6.0

Mastodon.**domain_blocks**(*max_id: str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *min_id: str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *since_id: str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *limit: int* | *None* = *None*) → *PaginatableList*[str]

Fetch the logged-in user's blocked domains.

Returns a list of blocked domain URLs (as strings, without protocol specifier).

Added: Mastodon v1.4.0, last changed: Mastodon v2.6.0

Mastodon.**account_mute**(*id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *notifications: bool* = *True*, *duration: int* | *None* = *None*) → *Relationship*

Mute a user.

Set *notifications* to False to receive notifications even though the user is muted from timelines. Pass a *duration* in seconds to have Mastodon automatically lift the mute after that many seconds.

The returned object reflects the updated relationship with the user.

Added: Mastodon v1.1.0, last changed: Mastodon v2.4.3 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**account_unmute**(*id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Relationship*

Unmute a user.

The returned object reflects the updated relationship with the user.

Added: Mastodon v1.1.0, last changed: Mastodon v1.4.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**account_block**(*id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Relationship*

Block a user.

The returned object reflects the updated relationship with the user.

Added: Mastodon v1.0.0, last changed: Mastodon v1.4.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**account_unblock**(*id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Relationship*

Unblock a user.

The returned object reflects the updated relationship with the user.

Added: Mastodon v1.0.0, last changed: Mastodon v1.4.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**account_remove_from_followers**(*id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Relationship*

Remove a user from the logged in users followers (i.e. make them unfollow the logged in user / "softblock" them).

The returned object reflects the updated relationship with the user.

Added: Mastodon v3.5.0, last changed: Mastodon v3.5.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**domain_block**(*domain: str*)

Add a block for all statuses originating from the specified domain for the logged-in user.

Added: Mastodon v1.4.0, last changed: Mastodon v1.4.0

Mastodon.**domain_unblock**(*domain: str*)

Remove a domain block for the logged-in user.

Added: Mastodon v1.4.0, last changed: Mastodon v1.4.0

Mastodon.**lists**() → *NonPaginatableList*[*UserList*]

Fetch a list of all the Lists by the logged-in user.

Added: Mastodon v2.1.0, last changed: Mastodon v2.1.0

Mastodon.**list**(*id:* UserList | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *UserList*

Fetch info about a specific list.

Added: Mastodon v2.1.0, last changed: Mastodon v2.1.0 (parameters), Mastodon v4.2.0 (return value)

Mastodon.**list_accounts**(*id:* UserList | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *max_id:* UserList | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *min_id:* UserList | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *since_id:* UserList | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *limit: int* | *None* = *None*) → *PaginatableList*[*Account*]

Get the accounts that are on the given list.

Added: Mastodon v2.1.0, last changed: Mastodon v2.6.0

Mastodon.**list_create**(*title: str*, *replies_policy: str* = *'list'*, *exclusive: bool* = *False*) → *UserList*

Create a new list with the given *title*.

replies_policy controls which replies are shown in the list. It can be one of "followed", "list" or "none". *followed* means that only replies from accounts you follow will be shown, *list* means that only replies from accounts on the list will be shown, and *none* means that no replies will be shown.

Set *exclusive* to True if you want the list to be an *exclusive* list, meaning that accounts on the list will be excluded from the home timeline, appearing exclusively in the list timeline.

Added: Mastodon v2.1.0, last changed: Mastodon v4.2.0 (parameters), Mastodon v4.2.0 (return value)

Mastodon.**list_update**(*id:* UserList | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *title: str*, *replies_policy: str* = *'list'*, *exclusive: bool* = *False*) → *UserList*

Update info about a list, where "info" is really the lists *title*.

replies_policy controls which replies are shown in the list. It can be one of "followed", "list" or "none". *followed* means that only replies from accounts you follow will be shown, *list* means that only replies from accounts on the list will be shown, and *none* means that no replies will be shown.

Set *exclusive* to True if you want the list to be an *exclusive* list, meaning that accounts on the list will be excluded from the home timeline, appearing exclusively in the list timeline.

The returned object reflects the updated list.

Added: Mastodon v2.1.0, last changed: Mastodon v4.2.0 (parameters), Mastodon v4.2.0 (return value)

`Mastodon.list_delete`(*id:* UserList | *str* | *int* | MaybeSnowflakeIdType | *datetime*)

Delete a list.

Added: Mastodon v2.1.0, last changed: Mastodon v2.1.0

`Mastodon.list_accounts_add`(*id:* UserList | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *account_ids:* *List[*Account | *str* | *int* | MaybeSnowflakeIdType | *datetime]*)

Add the account(s) given in *account_ids* to the list.

Added: Mastodon v2.1.0, last changed: Mastodon v2.1.0

`Mastodon.list_accounts_delete`(*id:* UserList | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *account_ids:* *List[*Account | *str* | *int* | MaybeSnowflakeIdType | *datetime]*)

Remove the account(s) given in *account_ids* from the list.

Added: Mastodon v2.1.0, last changed: Mastodon v2.1.0

`Mastodon.followed_tags`(*max_id:* Tag | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *min_id:* Tag | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *since_id:* Tag | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *limit: int* | *None* = *None*) → *PaginatableList*[*Tag*]

Returns the logged-in user's followed tags.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0

`Mastodon.tag_follow`(*hashtag:* Tag | *str*) → *Tag*

Follow a tag.

Returns the newly followed tag.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0 (parameters), Mastodon v4.4.0 (return value)

`Mastodon.tag_unfollow`(*hashtag:* Tag | *str*) → *Tag*

Unfollow a tag.

Returns the previously followed tag.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0 (parameters), Mastodon v4.4.0 (return value)

`Mastodon.timeline`(*timeline: str = 'home'*, *max_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *min_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *since_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *limit: int* | *None* = *None*, *only_media: bool* = *False*, *local: bool* = *False*, *remote: bool* = *False*) → *PaginatableList*[*Status*]

Fetch statuses, most recent ones first. *timeline* can be 'home', 'local', 'public', 'tag/<hashtag<', 'list/<id>' or 'link/<url>'. See the following functions documentation for what those do.

The default timeline is the "home" timeline.

Specify *only_media* to only get posts with attached media. Specify *local* to only get local statuses, and *remote* to only get remote statuses. Some options are mutually incompatible as dictated by logic.

May or may not require authentication depending on server settings and what is specifically requested.

See <https://docs.joinmastodon.org/methods/timelines/> for a description of the parameters.

Added: Mastodon v1.0.0, last changed: Mastodon v3.1.4

Mastodon.**timeline_home**(*max_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *min_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *since_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *limit: int* | *None* = *None*, *only_media: bool* = *False*, *local: bool* = *False*, *remote: bool* = *False*) → *PaginatableList*[*Status*]

Convenience method: Fetches the logged-in user's home timeline (i.e. followed users and self). Params as in *timeline()*.

Added: Mastodon v1.0.0, last changed: Mastodon v3.1.4

Mastodon.**timeline_local**(*max_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *min_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *since_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *limit: int* | *None* = *None*, *only_media: bool* = *False*) → *PaginatableList*[*Status*]

Convenience method: Fetches the local / instance-wide timeline, not including replies. Params as in *timeline()*.

Added: Mastodon v1.0.0, last changed: Mastodon v3.1.4

Mastodon.**timeline_public**(*max_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *min_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *since_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *limit: int* | *None* = *None*, *only_media: bool* = *False*, *local: bool* = *False*, *remote: bool* = *False*) → *PaginatableList*[*Status*]

Convenience method: Fetches the public / visible-network / federated timeline, not including replies. Params as in *timeline()*.

Added: Mastodon v1.0.0, last changed: Mastodon v3.1.4

Mastodon.**timeline_hashtag**(*hashtag: str*, *local: bool* = *False*, *max_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *min_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *since_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *limit: int* | *None* = *None*, *only_media: bool* = *False*, *remote: bool* = *False*) → *PaginatableList*[*Status*]

Convenience method: Fetch a timeline of toots with a given hashtag. The hashtag parameter should not contain the leading #. Params as in *timeline()*.

Added: Mastodon v1.0.0, last changed: Mastodon v3.1.4

Mastodon.**timeline_list**(*id:* UserList | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *max_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *min_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *since_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *limit: int* | *None* = *None*, *only_media: bool* = *False*, *local: bool* = *False*, *remote: bool* = *False*) → *PaginatableList*[*Status*]

Convenience method: Fetches a timeline containing all the toots by users in a given list. Params as in *timeline()*.

Added: Mastodon v2.1.0, last changed: Mastodon v3.1.4

Mastodon.**conversations**(*max_id:* Conversation | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *min_id:* Conversation | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *since_id:* Conversation | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None* = *None*, *limit: int* | *None* = *None*) → *PaginatableList*[*Conversation*]

Fetches a user's conversations.

Added: Mastodon v2.6.0, last changed: Mastodon v2.6.0

Mastodon.**instance**() → *InstanceV2* | *Instance*

Retrieve basic information about the instance, including the URI and administrative contact email.

Does not require authentication unless locked down by the administrator.

Will return the latest available version of the instance information. If you want a specific one, call the _v1 or _v2 variants

Added: Mastodon v1.1.0, last changed: Mastodon v4.0.0

Mastodon.**instance_v1**() → *Instance*

Retrieve basic information about the instance, including the URI and administrative contact email.

Does not require authentication unless locked down by the administrator.

This is the explicit v1 version of this function. The v2 version is available through instance_v2(). It contains a bit more information than this one, but does not include whether invites are enabled.

Added: Mastodon v1.1.0, last changed: Mastodon v2.3.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**instance_v2**() → *InstanceV2*

Retrieve basic information about the instance, including the URI and administrative contact email.

Does not require authentication unless locked down by the administrator. This is the explicit v2 variant.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0 (parameters), Mastodon v4.3.0 (return value)

Mastodon.**instance_activity**() → *NonPaginatableList*[*Activity*]

Retrieve activity stats about the instance. May be disabled by the instance administrator - throws a MastodonNotFoundError in that case.

Activity is returned for 12 weeks going back from the current week.

Added: Mastodon v2.1.2, last changed: Mastodon v2.1.2

Mastodon.**instance_peers**() → *NonPaginatableList*[str]

Retrieve the instances that this instance knows about. May be disabled by the instance administrator - throws a MastodonNotFoundError in that case.

Returns a list of URL strings.

Added: Mastodon v2.1.2, last changed: Mastodon v2.1.2

Mastodon.**instance_health**() → bool

Basic health check. Returns True if healthy, False if not.

Added: Mastodon v3.0.0, last changed: Mastodon v3.0.0

Mastodon.**instance_nodeinfo**(*schema: str = 'http://nodeinfo.diaspora.software/ns/schema/2.0'*) → *Nodeinfo*

Retrieves the instance's nodeinfo information.

For information on what the nodeinfo can contain, see the nodeinfo specification: https://github.com/jhass/nodeinfo . By default, Mastodon.py will try to retrieve the version 2.0 schema nodeinfo, for which we have a well defined return object. If you go outside of that, all bets are off.

To override the schema, specify the desired schema with the *schema* parameter.

Added: Mastodon v3.0.0, last changed: Mastodon v3.0.0 (parameters), Mastodon v3.0.0 (return value)

Mastodon.**instance_rules**() → *NonPaginatableList*[*Rule*]

Retrieve instance rules.

Added: Mastodon v3.4.0, last changed: Mastodon v3.4.0

Mastodon.**instance_extended_description**() → *ExtendedDescription*

Retrieve the instance's extended description.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**directory**(*offset: int | None = None*, *limit: int | None = None*, *order: str | None = None*, *local: bool | None = None*) → *NonPaginatableList*[*Account*]

Fetch the contents of the profile directory, if enabled on the server.

offset how many accounts to skip before returning results. Default 0.

limit how many accounts to load. Default 40.

order **"active" to sort by most recently posted statuses (usually the default) or**
"new" to sort by most recently created profiles.

local True to return only local accounts.

Uses offset/limit pagination, not currently handled by the pagination utility functions, do it manually if you have to.

Added: Mastodon v3.0.0, last changed: Mastodon v3.0.0

Mastodon.**custom_emojis**() → *NonPaginatableList*[*CustomEmoji*]

Fetch the list of custom emoji the instance has installed.

Does not require authentication unless locked down by the administrator.

Added: Mastodon v2.1.0, last changed: Mastodon v2.1.0

Mastodon.**announcements**() → *NonPaginatableList*[*Announcement*]

Fetch currently active announcements.

Added: Mastodon v3.1.0, last changed: Mastodon v3.1.0

Mastodon.**announcement_dismiss**(*id:* Announcement | *str* | *int* | MaybeSnowflakeIdType | *datetime*)

Set the given annoucement to read.

Added: Mastodon v3.1.0, last changed: Mastodon v3.1.0

Mastodon.**announcement_reaction_create**(*id:* Announcement | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *reaction: str*)

Add a reaction to an announcement. *reaction* can either be a unicode emoji or the name of one of the instances custom emoji.

Will throw an API error if the reaction name is not one of the allowed things or when trying to add a reaction that the user has already added (adding a reaction that a different user added is legal and increments the count).

Added: Mastodon v3.1.0, last changed: Mastodon v3.1.0

Mastodon.**announcement_reaction_delete**(*id:* Announcement | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *reaction: str*)

Remove a reaction to an announcement.

Will throw an API error if the reaction does not exist.

Added: Mastodon v3.1.0, last changed: Mastodon v3.1.0

Mastodon.**trending_tags**(*limit: int | None = None*, *offset: int | None = None*, *lang: str | None = None*) → *NonPaginatableList*[*Tag*]

Fetch trending-hashtag information, if the instance provides such information.

Specify *limit* to limit how many results are returned (default 10, the maximum number of results is 20).

Specify *offset* to paginate results. Default 0.

Does not require authentication unless locked down by the administrator.

Important versioning note: This endpoint does not exist for Mastodon versions between 2.8.0 (inclusive) and 3.0.0 (exclusive).

Pass *lang* to override the global locale parameter, which may affect trend ordering.

The results are sorted by the instances's trending algorithm, descending.

Added: Mastodon v3.5.0, last changed: Mastodon v3.5.0

Mastodon.**trending_statuses**(*limit: int | None = None*, *offset: int | None = None*, *lang: str | None = None*) → *NonPaginatableList*[*Status*]

Fetch trending-status information, if the instance provides such information.

Specify *limit* to limit how many results are returned (default 20, the maximum number of results is 40).

Specify *offset* to paginate results. Default 0.

Pass *lang* to override the global locale parameter, which may affect trend ordering.

The results are sorted by the instances's trending algorithm, descending.

Added: Mastodon v3.5.0, last changed: Mastodon v3.5.0

Mastodon.**trending_links**(*limit: int | None = None*, *offset: int | None = None*, *lang: str | None = None*) → *NonPaginatableList*[*PreviewCard*]

Fetch trending-link information, if the instance provides such information.

Specify *limit* to limit how many results are returned (default 10, the maximum number of results is 20).

Specify *offset* to paginate results. Default 0.

The results are sorted by the instances's trending algorithm, descending.

Added: Mastodon v3.5.0, last changed: Mastodon v3.5.0

Mastodon.**trends**(*limit: int | None = None*)

Old alias for *trending_tags()*

Deprecated. Please use *trending_tags()* instead.

Added: Mastodon v2.4.3, last changed: Mastodon v3.5.0

Mastodon.**search**(*q: str*, *resolve: bool = True*, *result_type: str | None = None*, *account_id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *offset: int | None = None*, *min_id: str | int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *max_id: str | int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *exclude_unreviewed: bool = True*) → *Search* | *SearchV2*

Fetch matching hashtags, accounts and statuses. Will perform webfinger lookups if resolve is True. Full-text search is only enabled if the instance supports it, and is restricted to statuses the logged-in user wrote or was mentioned in.

result_type can be one of "accounts", "hashtags" or "statuses", to only search for that type of object.

Specify *account_id* to only get results from the account with that id.

offset, *min_id* and *max_id* can be used to paginate.

exclude_unreviewed can be used to restrict search results for hashtags to only those that have been reviewed by moderators. It is on by default. When using the v1 search API (pre 2.4.1), it is ignored.

Will use search_v1 (no tags in return values) on Mastodon versions before 2.4.1), search_v2 otherwise. Parameters other than resolve are only available on Mastodon 2.8.0 or above - this function will throw a MastodonVersionError if you try to use them on versions before that. Note that the cached version number will be used for this to avoid uneccesary requests.

Added: Mastodon v1.1.0, last changed: Mastodon v2.8.0

Mastodon.**search_v2**(*q*, *resolve: bool = True*, *result_type: str | None = None*, *account_id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *offset: int | None = None*, *min_id: str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *max_id: str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *exclude_unreviewed: bool = True*) → *SearchV2*

Identical to *search_v1()*, except in that it returns tags as objects, has more parameters, and resolves by default.

For more details documentation, please see *search()*

Added: Mastodon v2.4.1, last changed: Mastodon v2.8.0 (parameters), Mastodon v2.4.1 (return value)

Mastodon.**instance_domain_blocks**() → *NonPaginatableList*[*DomainBlock*]

Fetch a list of domains that have been blocked by the instance. Public endpoint, requires authentication if limited to users.

Returns a MastodonAPIError if the admin has chosen to not make the list public, or to now show it at all.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0

Mastodon.**instance_languages**() → *NonPaginatableList*[*SupportedLocale*]

Fetch a list of languages that the instance supports.

Added: Mastodon v4.2.0, last changed: Mastodon v4.2.0

Mastodon.**instance_translation_languages**() → Dict[str, List[str]]

Retrieve the instance's translation languages.

Returns a dict with language pairs, where the key is the language code and the value is a list of language codes that the instance can translate that language to.

Mastodon.**notifications**(*id:* Notification | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *account_id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *max_id:* Notification | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *min_id:* Notification | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *since_id:* Notification | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *limit: int | None = None*, *exclude_types: List[str] | None = None*, *types: List[str] | None = None*, *mentions_only: bool | None = None*) → *PaginatableList*[*Notification*] | *Notification*

Fetch notifications (mentions, favourites, reblogs, follows) for the logged-in user. Pass *account_id* to get only notifications originating from the given account.

There are different types of notifications:

- *follow* - A user followed the logged in user
- *follow_request* - A user has requested to follow the logged in user (for locked accounts)
- *favourite* - A user favourited a post by the logged in user
- *reblog* - A user reblogged a post by the logged in user
- *mention* - A user mentioned the logged in user

- *poll* - A poll the logged in user created or voted in has ended
- *update* - A status the logged in user has reblogged (and only those, as of 4.0.0) has been edited
- *status* - A user that the logged in user has enabned notifications for has enabled *notify* (see *account_follow()*)
- *admin.sign_up* - For accounts with appropriate permissions: A new user has signed up
- *admin.report* - For accounts with appropriate permissions: A new report has been received
- *severed_relationships* - Some of the logged in users relationships have been severed due to a moderation action on this server
- *moderation_warning* - The logged in user has been warned by a moderator

Parameters *exclude_types* and *types* are array of these types, specifying them will in- or exclude the types of notifications given. It is legal to give both parameters at the same tine, the result will then be the intersection of the results of both filters. Specifying *mentions_only* is a deprecated way to set *exclude_types* to all but mentions.

Can be passed an *id* to fetch a single notification.

Added: Mastodon v1.0.0, last changed: Mastodon v3.5.0

Mastodon.**notifications_unread_count**() → *UnreadNotificationsCount*

Fetch the number of unread notifications for the logged-in user.

Added: Mastodon v4.3.0, last changed: Mastodon v4.3.0 (parameters), Mastodon v4.3.0 (return value)

Mastodon.**notifications_clear**()

Clear out a user's notifications

Added: Mastodon v1.0.0, last changed: Mastodon v1.0.0

Mastodon.**notifications_dismiss**(*id:* Notification | *str* | *int* | MaybeSnowflakeIdType | *datetime*)

Deletes a single notification

Added: Mastodon v1.3.0, last changed: Mastodon v2.9.2

Mastodon.**conversations_read**(*id:* Conversation | *str* | *int* | MaybeSnowflakeIdType | *datetime*)

Marks a single conversation as read.

The returned object reflects the conversation's new read status.

Added: Mastodon v2.6.0, last changed: Mastodon v2.6.0

Mastodon.**grouped_notifications**(*max_id: str | int | MaybeSnowflakeIdType | datetime | None = None, since_id: str | int | MaybeSnowflakeIdType | datetime | None = None, min_id: str | int | MaybeSnowflakeIdType | datetime | None = None, limit: int | None = None, types: List[str] | None = None, exclude_types: List[str] | None = None, account_id:* Account | *str | int | MaybeSnowflakeIdType | datetime | None = None, expand_accounts: str | None = 'partial_avatars', grouped_types: List[str] | None = None, include_filtered: bool | None = None*) → *GroupedNotificationsResults*

Fetch grouped notifications for the user. Requires scope *read:notifications*.

For base parameters, see *notifications()*.

grouped_types controls which notication types can be grouped together - all, if not specified. NB: "all" here means favourite, follow and reblog - other types are not groupable and are returned individually (with a unique group key) always.

Pass *include_filtered=True* to include filtered notifications in the response.

Pass *expand_accounts="full"* to include full account details in the response, or "partial_avatars" to include a smaller set of account details (in the *partial_accounts* field) for some (but not all - the most recent account triggering a notification is always returned in full) of the included accounts. The default is partial_avatars.

Added: Mastodon v4.3.0, last changed: Mastodon v4.3.0 (parameters), Mastodon v4.3.0 (return value)

Mastodon.**grouped_notification**(*group_key: str*) → *GroupedNotificationsResults*

Fetch details of a single grouped notification by its group key. Requires scope *read:notifications*.

Added: Mastodon v4.3.0, last changed: Mastodon v4.3.0 (parameters), Mastodon v4.3.0 (return value)

Mastodon.**dismiss_grouped_notification**(*group_key: str*) → None

Dismiss a single grouped notification. Requires scope *write:notifications*.

Added: Mastodon v4.3.0, last changed: Mastodon v4.3.0

Mastodon.**grouped_notification_accounts**(*group_key: str*) → *NonPaginatableList*[*Account*]

Fetch accounts associated with a grouped notification. Requires scope *write:notifications*.

Added: Mastodon v4.3.0, last changed: Mastodon v4.3.0

Mastodon.**unread_grouped_notifications_count**(*limit: int | None = None*, *types: List[str] | None = None*, *exclude_types: List[str] | None = None*, *account_id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *grouped_types: List[str] | None = None*) → int

Fetch the count of unread grouped notifications. Requires scope *read:notifications*.

For parameters, see *notifications()* and *grouped_notifications()*.

Added: Mastodon v4.3.0, last changed: Mastodon v4.3.0

Mastodon.**notifications_policy**() → *NotificationPolicy*

Fetch the user's notification filtering policy. Requires scope *read:notifications*.

Added: Mastodon v4.3.0, last changed: Mastodon v4.3.0 (parameters), Mastodon v4.3.0 (return value)

Mastodon.**update_notifications_policy**(*for_not_following: str | None = None*, *for_not_followers: str | None = None*, *for_new_accounts: str | None = None*, *for_private_mentions: str | None = None*, *for_limited_accounts: str | None = None*) → *NotificationPolicy*

Update the user's notification filtering policy. Requires scope *write:notifications*.

- *for_not_following*: "accept", "filter", or "drop" notifications from non-followed accounts.
- *for_not_followers*: "accept", "filter", or "drop" notifications from non-followers.
- *for_new_accounts*: "accept", "filter", or "drop" notifications from accounts created in the past 30 days.
- *for_private_mentions*: "accept", "filter", or "drop" notifications from private mentions.
- *for_limited_accounts*: "accept", "filter", or "drop" notifications from accounts limited by moderators.

Added: Mastodon v4.3.0, last changed: Mastodon v4.3.0 (parameters), Mastodon v4.3.0 (return value)

Mastodon.**notification_requests**(*max_id: str | int |* MaybeSnowflakeIdType | *datetime | None = None*, *since_id: str | int |* MaybeSnowflakeIdType | *datetime | None = None*, *min_id: str | int |* MaybeSnowflakeIdType | *datetime | None = None*, *limit: int | None = None*) → *PaginatableList*[*NotificationRequest*]

Fetch notification requests filtered by the user's policy. Requires scope *read:notifications*.

NB: Notification requests are what happens when the user has set their policy to filter notifications from some source.

Added: Mastodon v4.3.0, last changed: Mastodon v4.3.0

Mastodon.**notification_request**(*id:* NotificationRequest | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *NotificationRequest*

Fetch a single notification request by ID. Requires scope *read:notifications*.

Added: Mastodon v4.3.0, last changed: Mastodon v4.3.0 (parameters), Mastodon v4.3.0 (return value)

Mastodon.**accept_notification_request**(*id:* NotificationRequest | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → None

Accept a notification request. This moves filtered notifications from a user back into the main notifications feed and allows future notifications from them. Requires scope *write:notifications*.

Added: Mastodon v4.3.0, last changed: Mastodon v4.3.0

Mastodon.**dismiss_notification_request**(*id:* NotificationRequest | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → None

Dismiss a notification request, removing it from pending requests. Requires scope *write:notifications*.

Added: Mastodon v4.3.0, last changed: Mastodon v4.3.0

Mastodon.**accept_multiple_notification_requests**(*ids: List[*NotificationRequest | *str* | *int* | MaybeSnowflakeIdType | *datetime]*) → None

Accept multiple notification requests at once. This moves filtered notifications from those users back into the main notifications feed and allows future notifications from them. Requires scope *write:notifications*.

Added: Mastodon v4.3.0, last changed: Mastodon v4.3.0

Mastodon.**dismiss_multiple_notification_requests**(*ids: List[*NotificationRequest | *str* | *int* | MaybeSnowflakeIdType | *datetime]*) → None

Dismiss multiple notification requests, removing them from pending requests. Requires scope *write:notifications*.

Added: Mastodon v4.3.0, last changed: Mastodon v4.3.0

Mastodon.**notifications_merged**() → bool

Check whether accepted notification requests have been merged into the main notification feed. Accepting a notification request schedules a background job that merges the filtered notifications. Clients can poll this endpoint to check if the merge has completed. Requires scope *read:notifications*.

Added: Mastodon v4.3.0, last changed: Mastodon v4.3.0

Mastodon.**filters_v2**() → *NonPaginatableList*[*FilterV2*]

Fetch all filters for the authenticated user.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0

Mastodon.**filter_v2**(*filter_id:* Filter | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Filter*

Fetch a specific filter by its ID.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0 (parameters), Mastodon v3.1.0 (return value)

Mastodon.**create_filter_v2**(*title: str*, *context: List[str]*, *filter_action: str*, *expires_in: int | None = None*, *keywords_attributes: List[Dict[str, str | bool]] | None = None*) → *FilterV2*

Create a new filter with the given parameters.

title is a human readable name for the filter.

***context* is list of contexts where the filter should apply. Valid values are:**

- "home": Filter applies to the home timeline.
- "notifications": Filter applies to notifications. Filtered notifications land in notification requests.

- "public": Filter applies to the public timelines.
- "thread": Filter applies to conversations.
- "account": Filter applies to account timelines.

***filter_action* gives the policy to be applied when the filter is matched. Valid values are:**

- "warn": The user is warned if the content matches the filter.
- "hide": The content is completely hidden if it matches the filter.

NB: Even if you specify "hide", the status will still be returned - it will just have the "filtered" attribute set.

pass a number of seconds as *expires_in* to make the filter expire in that many seconds. Use None for no expiration.

pass a list of keyword dicts to initially as *keywords_attributes*, each with the following values:

- "keyword": The term to filter on.
- "whole_word": Whether word boundaries should be considered.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0 (parameters), Mastodon v4.0.0 (return value)

`Mastodon.`**`update_filter_v2`**(*filter_id:* FilterV2 | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *title: str* | *None* = *None*, *context: List[str]* | *None* = *None*, *filter_action: str* | *None* = *None*, *expires_in: int* | *None* = *None*, *keywords_attributes: List[Dict[str, str* | *bool* | *int]]* | *None* = *None*) → *FilterV2*

Update an existing filter with the given parameters.

Parameters are as in *create_filter_v2()*. Only the parameters you want to update need to be provided.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0 (parameters), Mastodon v4.0.0 (return value)

`Mastodon.`**`delete_filter_v2`**(*filter_id:* FilterV2 | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → None

Delete an existing filter.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0

`Mastodon.`**`filter_keywords_v2`**(*filter_id:* FilterV2 | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *NonPaginatableList*[*FilterKeyword*]

Fetch all keywords associated with a given filter.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0

`Mastodon.`**`add_filter_keyword_v2`**(*filter_id:* FilterV2 | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *keyword: str*, *whole_word: bool* = *False*) → *FilterKeyword*

Add a single keyword to an existing filter.

Parameters are as in *create_filter_v2() keywords_attributes*.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0 (parameters), Mastodon v4.0.0 (return value)

`Mastodon.`**`delete_filter_keyword_v2`**(*keyword_id:* FilterKeyword | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → None

Delete a single keyword from any filter.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0

`Mastodon.`**`filter_statuses_v2`**(*filter_id:* FilterV2 | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → List[*FilterStatus*]

Retrieve all status-based filters for a FilterV2.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0

Mastodon.**add_filter_status_v2**(*filter_id:* FilterV2 | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *status_id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *FilterStatus*

Add a status to a filter, which will then match on that status in addition to any keywords. Includes reblogs, does not include replies.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**filter_status_v2**(*filter_status_id:* FilterStatus | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *FilterStatus*

Fetch a single status-based filter by its ID.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**delete_filter_status_v2**(*filter_status_id:* FilterStatus | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → None

Remove a status filter from a FilterV2.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0

Mastodon.**push_subscription**() → *WebPushSubscription*

Fetch the current push subscription the logged-in user has for this app.

Only one webpush subscription can be active at a time for any given app.

Added: Mastodon v2.4.0, last changed: Mastodon v2.4.0 (parameters), Mastodon v4.4.0 (return value)

Mastodon.**push_subscription_set**(*endpoint: str*, *encrypt_params: Dict[str, str]*, *follow_events: bool | None = None*, *favourite_events: bool | None = None*, *reblog_events: bool | None = None*, *mention_events: bool | None = None*, *poll_events: bool | None = None*, *follow_request_events: bool | None = None*, *status_events: bool | None = None*, *policy: str = 'all'*, *update_events: bool | None = None*, *admin_sign_up_events: bool | None = None*, *admin_report_events: bool | None = None*, *standard: bool = None*) → *WebPushSubscription*

Sets up or modifies the push subscription the logged-in user has for this app.

endpoint is the endpoint URL mastodon should call for pushes. Note that mastodon requires https for this URL. *encrypt_params* is a dict with key parameters that allow the server to encrypt data for you: A public key *pubkey* and a shared secret *auth*. You can generate this as well as the corresponding private key using the *push_subscription_generate_keys()* function.

policy controls what sources will generate webpush events. Valid values are *all*, *none*, *follower* and *followed*.

The rest of the parameters controls what kind of events you wish to subscribe to. Events whose names start with "admin" require admin privileges to subscribe to.

- *follow_events* controls whether you receive events when someone follows the logged in user.
- *favourite_events* controls whether you receive events when someone favourites one of the logged in users statuses.
- *reblog_events* controls whether you receive events when someone boosts one of the logged in users statuses.
- *mention_events* controls whether you receive events when someone mentions the logged in user in a status.
- *poll_events* controls whether you receive events when a poll the logged in user has voted in has ended.
- *follow_request_events* controls whether you receive events when someone requests to follow the logged in user.
- *status_events* controls whether you receive events when someone the logged in user has subscribed to notifications for posts a new status.

- *update_events* controls whether you receive events when a status that the logged in user has boosted has been edited.
- *admin_sign_up_events* controls whether you receive events when a new user signs up.
- *admin_report_events* controls whether you receive events when a new report is received.

Pass *standard=True* to use the standard webpush subscription format, instead of the pre-release RFC format mastodon was using before.

Added: Mastodon v2.4.0, last changed: Mastodon v4..0 (parameters), Mastodon v4.4.0 (return value)

Mastodon.**push_subscription_update**(*follow_events: bool | None = None*, *favourite_events: bool | None = None*, *reblog_events: bool | None = None*, *mention_events: bool | None = None*, *poll_events: bool | None = None*, *follow_request_events: bool | None = None*, *status_events: bool | None = None*, *policy: str | None = 'all'*, *update_events: bool | None = None*, *admin_sign_up_events: bool | None = None*, *admin_report_events: bool | None = None*) → *WebPushSubscription*

Modifies what kind of events the app wishes to subscribe to.

Parameters are as in push_subscription_set().

Returned object reflects the updated push subscription.

Added: Mastodon v2.4.0, last changed: Mastodon v2.4.0 (parameters), Mastodon v4.4.0 (return value)

Mastodon.**push_subscription_generate_keys**() → Tuple[Dict[str, str], Dict[str, str]]

Generates a private key, public key and shared secret for use in webpush subscriptions.

Returns two dicts: One with the private key and shared secret and another with the public key and shared secret.

Mastodon.**push_subscription_decrypt_push**(*data: bytes*, *decrypt_params: Dict[str, str]*, *encryption_header: str*, *crypto_key_header: str*) → *PushNotification*

Decrypts *data* received in a webpush request. Requires the private key dict from *push_subscription_generate_keys()* (*decrypt_params*) as well as the Encryption and server Crypto-Key headers from the received webpush

Added: Mastodon v2.4.0, last changed: Mastodon v2.4.0 (parameters), Mastodon v2.4.0 (return value)

Mastodon.**filters**() → *NonPaginatableList*[*Filter*]

Fetch all of the logged-in user's filters.

Added: Mastodon v2.4.3, last changed: Mastodon v2.4.3

Mastodon.**filter**(*id:* Filter | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Filter*

Fetches information about the filter with the specified *id*.

Added: Mastodon v2.4.3, last changed: Mastodon v2.4.3 (parameters), Mastodon v3.1.0 (return value)

Mastodon.**filters_apply**(*objects:* PaginatableList*[*Status*]* | PaginatableList*[*Notification*]*, *filters:* NonPaginatableList*[*Filter*]* | NonPaginatableList*[*FilterV2*]*, *context: str*) → *PaginatableList*[*Status*] | *PaginatableList*[*Notification*]

Helper function: Applies a list of filters to a list of either statuses or notifications and returns only those matched by none. This function will apply all filters that match the context provided in *context*, i.e. if you want to apply only notification-relevant filters, specify 'notifications'. Valid contexts are 'home', 'notifications', 'public' and 'thread'.

NB: This is for v1 filters. v2 filters are applied by the server, which adds the "filtered" attribute to filtered statuses.

Added: Mastodon v2.4.3, last changed: Mastodon v2.4.3

Mastodon.**filter_create**(*phrase: str*, *context: str*, *irreversible: bool = False*, *whole_word: bool = True*, *expires_in: int | None = None*) → *Filter*

Creates a new keyword filter. *phrase* is the phrase that should be filtered out, *context* specifies from where to filter the keywords. Valid contexts are 'home', 'notifications', 'public' and 'thread'.

Set *irreversible* to True if you want the filter to just delete statuses server side. This works only for the 'home' and 'notifications' contexts.

Set *whole_word* to False if you want to allow filter matches to start or end within a word, not only at word boundaries.

Set *expires_in* to specify for how many seconds the filter should be kept around.

Returns the newly created filter.

Added: Mastodon v2.4.3, last changed: Mastodon v2.4.3 (parameters), Mastodon v3.1.0 (return value)

Mastodon.**filter_update**(*id: Filter | str | int | MaybeSnowflakeIdType | datetime*, *phrase: str | None = None*, *context: str | None = None*, *irreversible: bool | None = None*, *whole_word: bool | None = None*, *expires_in: int | None = None*) → *Filter*

Updates the filter with the given *id*. Parameters are the same as in *filter_create()*.

Returns the updated filter.

Added: Mastodon v2.4.3, last changed: Mastodon v2.4.3 (parameters), Mastodon v3.1.0 (return value)

Mastodon.**filter_delete**(*id: Filter | FilterV2 | str | int | MaybeSnowflakeIdType | datetime*)

Deletes the filter with the given *id*.

Added: Mastodon v2.4.3, last changed: Mastodon v2.4.3

Mastodon.**stream_user**(*listener*, *run_async=False*, *timeout=300*, *reconnect_async=False*, *reconnect_async_wait_sec=5*)

Streams events that are relevant to the authorized user, i.e. home timeline and notifications.

Added: Mastodon v1.1.0, last changed: Mastodon v1.4.2

Mastodon.**stream_public**(*listener*, *run_async=False*, *timeout=300*, *reconnect_async=False*, *reconnect_async_wait_sec=5*, *local=False*, *remote=False*)

Streams public events.

Set *local* to True to only get local statuses. Set *remote* to True to only get remote statuses.

Added: Mastodon v1.1.0, last changed: Mastodon v1.4.2

Mastodon.**stream_local**(*listener*, *run_async=False*, *timeout=300*, *reconnect_async=False*, *reconnect_async_wait_sec=5*)

Streams local public events.

This function is deprecated. Please use stream_public() with parameter *local* set to True instead.

Added: Mastodon v1.1.0, last changed: Mastodon v1.4.2

Mastodon.**stream_hashtag**(*tag*, *listener*, *local=False*, *run_async=False*, *timeout=300*, *reconnect_async=False*, *reconnect_async_wait_sec=5*)

Stream for all public statuses for the hashtag 'tag' seen by the connected instance.

Set *local* to True to only get local statuses.

Added: Mastodon v1.1.0, last changed: Mastodon v1.4.2

Mastodon.**stream_list**(*id*, *listener*, *run_async=False*, *timeout=300*, *reconnect_async=False*, *reconnect_async_wait_sec=5*)

Stream events for the current user, restricted to accounts on the given list.

Added: Mastodon v2.1.0, last changed: Mastodon v2.1.0

Mastodon.**stream_healthy**() → bool

Returns True if streaming API is okay, False or raises an error otherwise.

Added: Mastodon v2.5.0, last changed: Mastodon v2.5.0

class mastodon.**StreamListener**

Callbacks for the streaming API. Create a subclass, override the on_xxx methods for the kinds of events you're interested in, then pass an instance of your subclass to Mastodon.user_stream(), Mastodon.public_stream(), or Mastodon.hashtag_stream().

StreamListener.**on_update**(*status:* Status)

A new status has appeared. *status* is the parsed *status dict* describing the status.

StreamListener.**on_notification**(*notification:* Notification)

A new notification. *notification* is the object describing the notification. For more information, see the documentation for the notifications() method.

StreamListener.**on_delete**(*status_id: str | int |* MaybeSnowflakeIdType *| datetime*)

A status has been deleted. *status_id* is the status' integer ID.

StreamListener.**on_conversation**(*conversation:* Conversation)

A direct message (in the direct stream) has been received. *conversation* is the parsed *conversation dict* dictionary describing the conversation

StreamListener.**on_status_update**(*status:* Status)

A status has been edited. 'status' is the parsed JSON dictionary describing the updated status.

StreamListener.**on_unknown_event**(*name*, *unknown_event=None*)

An unknown mastodon API event has been received. The name contains the event-name and unknown_event contains the content of the unknown event.

StreamListener.**on_abort**(*err*)

There was a connection error, read timeout or other error fatal to the streaming connection. The exception object about to be raised is passed to this function for reference.

Note that the exception will be raised properly once you return from this function, so if you are using this handler to reconnect, either never return or start a thread and then catch and ignore the exception.

StreamListener.**handle_heartbeat**()

The server has sent us a keep-alive message. This callback may be useful to carry out periodic housekeeping tasks, or just to confirm that the connection is still open.

class mastodon.**CallbackStreamListener**(*update_handler=None*, *local_update_handler=None*, *delete_handler=None*, *notification_handler=None*, *conversation_handler=None*, *unknown_event_handler=None*, *status_update_handler=None*, *filters_changed_handler=None*, *announcement_handler=None*, *announcement_reaction_handler=None*, *announcement_delete_handler=None*, *encryted_message_handler=None*)

Simple callback stream handler class. Can optionally additionally send local update events to a separate handler. Define an unknown_event_handler for new Mastodon API events. This handler is *not* guaranteed to receive these events forever, and should only be used for diagnostics.

Mastodon.**markers_get**(*timeline: str | List[str] = ['home']*) → Dict[str, *Marker*]

Get the last-read-location markers for the specified timelines. Valid timelines are *home* (the home timeline) and *notifications* (the notifications timeline, affects which notifications are considered read).

Note that despite the singular name, *timeline* can be a list.

Returns a dict with the markers, keyed by timeline name.

Added: Mastodon v3.0.0, last changed: Mastodon v3.0.0

Mastodon.**markers_set**(*timelines: str | List[str], last_read_ids:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *List[*Status*]* | *List[str* | *int* | MaybeSnowflakeIdType | *datetime]*) → Dict[str, *Marker*]

Set the "last read" marker(s) for the given timeline(s) to the given id(s)

Valid timelines are *home* (the home timeline) and *notifications* (the notifications timeline, affects which notifications are considered read).

Note that if you give an invalid timeline name, this will silently do nothing.

Returns a dict with the updated markers, keyed by timeline name.

Added: Mastodon v3.0.0, last changed: Mastodon v3.0.0

Mastodon.**reports**() → *NonPaginatableList*[*Report*]

Fetch a list of reports made by the logged-in user.

Warning: This method has now finally been removed, and will not work on Mastodon versions 2.5.0 and above.

Added: Mastodon v1.1.0, last changed: Mastodon v1.1.0

Mastodon.**report**(*account_id:* Account | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *status_ids:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *comment: str | None = None*, *forward: bool = False*, *category: str | None = None*, *rule_ids: List[*Rule | *str* | *int* | MaybeSnowflakeIdType | *datetime]* | *None = None*, *forward_to_domains: List[str] | None = None*) → *Report*

Report statuses to the instances administrators.

Accepts a list of toot IDs associated with the report, and a comment.

Starting with Mastodon 3.5.0, you can also pass a *category* (one out of "spam", "violation" or "other") and *rule_ids* (a list of rule IDs corresponding to the rules returned by the *instance()* API).

Set *forward* to True to forward a report of a remote user to that users instance as well as sending it to the instance local administrators. Set forward_to_domains to a list of domains to forward the report to (only domains of people mentioned in the status), or omitto forward to the domain of the reported status.

Added: Mastodon v1.1.0, last changed: Mastodon v3.5.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**fetch_next**(*previous_page:* PaginatableList*[*Entity*]* | Entity | *PaginationInfo*) → *PaginatableList*[*Entity*] | *Entity* | None

Fetches the next page of results of a paginated request. Pass in the previous page in its entirety, or the pagination information dict returned as a part of that pages last status ('_pagination_next').

Returns the next page or None if no further data is available.

Mastodon.**fetch_previous**(*next_page:* PaginatableList*[*Entity*]* | Entity | *PaginationInfo*) → *PaginatableList*[*Entity*] | *Entity* | None

Fetches the previous page of results of a paginated request. Pass in the previous page in its entirety, or the pagination information dict returned as a part of that pages first status ('_pagination_prev').

Returns the previous page or None if no further data is available.

Mastodon.**fetch_remaining**(*first_page:* PaginatableList*[*Entity*]*) → *PaginatableList*[*Entity*]

Fetches all the remaining pages of a paginated request starting from a first page and returns the entire set of results (including the first page that was passed in) as a big list.

Be careful, as this might generate a lot of requests, depending on what you are fetching, and might cause you to run into rate limits very quickly.

Does not work with grouped notifications, since they use a somewhat weird, inside-out pagination scheme. If you need to access these in a paginated way, use fetch_next and fetch_previous directly.

Mastodon.**decode_blurhash**(*media_dict:* MediaAttachment, *out_size: Tuple[int, int] = (16, 16)*, *size_per_component: bool = True*, *return_linear: bool = True*) → List[List[List[float]]]

Basic media-dict blurhash decoding.

out_size is the desired result size in pixels, either absolute or per blurhash component (this is the default).

By default, this function will return the image as linear RGB, ready for further scaling operations. If you want to display the image directly, set return_linear to False.

Returns the decoded blurhash image as a three-dimensional list: [height][width][3], with the last dimension being RGB colours.

For further info and tips for advanced usage, refer to the documentation for the blurhash module: https://github.com/halcy/blurhash-python

Mastodon.**clear_caches**()

Clear cached data for astodon version and streaming base URL. Most programs should not have to call this.

Mastodon.**get_approx_server_time**() → <module 'datetime' from '/home/docs/.asdf/installs/python/3.12.10/lib/python3.12/datetime.py'>

Retrieve the approximate server time

We parse this from the hopefully present "Date" header, but make no effort to compensate for latency.

Mastodon.**admin_accounts_v2**(*origin: str | None = None*, *by_domain: str | None = None*, *status: str | None = None*, *username: str | None = None*, *display_name: str | None = None*, *email: str | None = None*, *ip: str | None = None*, *permissions: str | None = None*, *invited_by:* Account *| str | int |* MaybeSnowflakeIdType *| datetime = None*, *role_ids: List[str | int |* MaybeSnowflakeIdType *| datetime] | None = None*, *max_id: str | int |* MaybeSnowflakeIdType *| datetime | None = None*, *min_id: str | int |* MaybeSnowflakeIdType *| datetime | None = None*, *since_id: str | int |* MaybeSnowflakeIdType *| datetime | None = None*, *limit: int | None = None*) → List[*AdminAccount*]

Fetches a list of accounts that match given criteria. By default, local accounts are returned.

- Set *origin* to "local" or "remote" to get only local or remote accounts.
- Set *by_domain* to a domain to get only accounts from that domain.
- Set *status* to one of "active", "pending", "disabled", "silenced" or "suspended" to get only accounts with that moderation status (default: active)
- Set *username* to a string to get only accounts whose username contains this string.
- Set *display_name* to a string to get only accounts whose display name contains this string.
- Set *email* to an email to get only accounts with that email (this only works on local accounts).
- Set *ip* to an ip (as a string, standard v4/v6 notation) to get only accounts whose last active ip is that ip (this only works on local accounts).

- Set *permissions* to "staff" to only get accounts with staff permissions.
- Set *invited_by* to an account id to get only accounts invited by this user.
- Set *role_ids* to a list of role IDs to get only accounts with those roles.

Pagination on this is a bit weird, so I would recommend not doing that and instead manually fetching.

Added: Mastodon v2.9.1, last changed: Mastodon v4.0.0

Mastodon.**admin_accounts**(*remote: bool = False*, *by_domain: str | None = None*, *status: str = 'active'*, *username: str | None = None*, *display_name: str | None = None*, *email: str | None = None*, *ip: str | None = None*, *staff_only: bool = False*, *max_id: str | int | MaybeSnowflakeIdType | datetime | None = None*, *min_id: str | int | MaybeSnowflakeIdType | datetime | None = None*, *since_id: str | int | MaybeSnowflakeIdType | datetime | None = None*, *limit: int | None = None*)

Currently a synonym for admin_accounts_v1, now deprecated. You are strongly encouraged to use admin_accounts_v2 instead, since this one is kind of bad.

!!!!! This function may be switched to calling the v2 API in the future. This is your warning. If you want to keep using v1, use it explicitly. !!!!!

Pagination on this is a bit weird, so I would recommend not doing that and instead manually fetching.

Added: Mastodon v2.9.1, last changed: Mastodon v2.9.1

Mastodon.**admin_accounts_v1**(*remote: bool = False*, *by_domain: str | None = None*, *status: str = 'active'*, *username: str | None = None*, *display_name: str | None = None*, *email: str | None = None*, *ip: str | None = None*, *staff_only: bool = False*, *max_id: str | int | MaybeSnowflakeIdType | datetime | None = None*, *min_id: str | int | MaybeSnowflakeIdType | datetime | None = None*, *since_id: str | int | MaybeSnowflakeIdType | datetime | None = None*, *limit: int | None = None*) → *AdminAccount*

Fetches a list of accounts that match given criteria. By default, local accounts are returned.

- Set *remote* to True to get remote accounts, otherwise local accounts are returned (default: local accounts)
- Set *by_domain* to a domain to get only accounts from that domain.
- Set *status* to one of "active", "pending", "disabled", "silenced" or "suspended" to get only accounts with that moderation status (default: active)
- Set *username* to a string to get only accounts whose username contains this string.
- Set *display_name* to a string to get only accounts whose display name contains this string.
- Set *email* to an email to get only accounts with that email (this only works on local accounts).
- Set *ip* to an ip (as a string, standard v4/v6 notation) to get only accounts whose last active ip is that ip (this only works on local accounts).
- Set *staff_only* to True to only get staff accounts (this only works on local accounts).

Note that setting the boolean parameters to False does not mean "give me users to which this does not apply" but instead means "I do not care if users have this attribute".

Deprecated in Mastodon version 3.5.0.

Pagination on this is a bit weird, so I would recommend not doing that and instead manually fetching.

Added: Mastodon v2.9.1, last changed: Mastodon v2.9.1 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**admin_account**(*id:* Account | AdminAccount | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *AdminAccount*

Fetches a single admin account for the user with the given id.

Added: Mastodon v2.9.1, last changed: Mastodon v2.9.1 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**admin_account_enable**(*id:* Account | AdminAccount | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *AdminAccount*

Reenables login for a local account for which login has been disabled.

The returned object reflects the updates to the account.

Added: Mastodon v2.9.1, last changed: Mastodon v2.9.1 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**admin_account_approve**(*id:* Account | AdminAccount | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *AdminAccount*

Approves a pending account.

The returned object reflects the updates to the account.

Added: Mastodon v2.9.1, last changed: Mastodon v2.9.1 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**admin_account_reject**(*id:* Account | AdminAccount | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *AdminAccount*

Rejects and deletes a pending account.

The returned object is that of the now-deleted account.

Added: Mastodon v2.9.1, last changed: Mastodon v2.9.1 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**admin_account_unsilence**(*id:* Account | AdminAccount | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *AdminAccount*

Unsilences an account.

The returned object reflects the updates to the account.

Added: Mastodon v2.9.1, last changed: Mastodon v2.9.1 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**admin_account_unsuspend**(*id:* Account | AdminAccount | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *AdminAccount*

Unsuspends an account.

The returned object reflects the updates to the account.

Added: Mastodon v2.9.1, last changed: Mastodon v2.9.1 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**admin_account_moderate**(*id:* Account | AdminAccount | *str* | *int* | MaybeSnowflakeIdType | *datetime*, *action: str* | *None* = *None*, *report_id:* AdminReport | *str* | *int* | *None* = *None*, *warning_preset_id: str* | *int* | *None* = *None*, *text: str* | *None* = *None*, *send_email_notification: bool* | *None* = *True*)

Perform a moderation action on an account.

Valid actions are:

- "disable" - for a local user, disable login.
- "silence" - hide the users posts from all public timelines.
- "suspend" - irreversibly delete all the user's posts, past and future.
- "sensitive" - forcce an accounts media visibility to always be sensitive.

If no action is specified, the user is only issued a warning.

Specify the id of a report as *report_id* to close the report with this moderation action as the resolution. Specify *warning_preset_id* to use a warning preset as the notification text to the user, or *text* to specify text directly. If both are specified, they are concatenated (preset first). Note that there is currently no API to retrieve or create warning presets.

Set *send_email_notification* to False to not send the user an email notification informing them of the moderation action.

Added: Mastodon v2.9.1, last changed: Mastodon v2.9.1

`Mastodon.admin_reports`(*resolved: bool | None = False*, *account_id:* Account | AdminAccount | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *target_account_id:* Account | AdminAccount | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *max_id: str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *min_id: str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *since_id: str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *limit: int* | *None = None*) → *PaginatableList*[*AdminReport*]

Fetches the list of reports.

Set *resolved* to True to search for resolved reports. *account_id* and *target_account_id* can be used to get reports filed by or about a specific user.

Added: Mastodon v2.9.1, last changed: Mastodon v2.9.1

`Mastodon.admin_report`(*id:* AdminReport | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *AdminReport*

Fetches the report with the given id.

Added: Mastodon v2.9.1, last changed: Mastodon v2.9.1 (parameters), Mastodon v4.0.0 (return value)

`Mastodon.admin_report_assign`(*id:* AdminReport | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *AdminReport*

Assigns the given report to the logged-in user.

The returned object reflects the updates to the report.

Added: Mastodon v2.9.1, last changed: Mastodon v2.9.1 (parameters), Mastodon v4.0.0 (return value)

`Mastodon.admin_report_unassign`(*id:* AdminReport | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *AdminReport*

Unassigns the given report from the logged-in user.

The returned object reflects the updates to the report.

Added: Mastodon v2.9.1, last changed: Mastodon v2.9.1 (parameters), Mastodon v4.0.0 (return value)

`Mastodon.admin_report_reopen`(*id:* AdminReport | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *AdminReport*

Reopens a closed report.

The returned object reflects the updates to the report.

Added: Mastodon v2.9.1, last changed: Mastodon v2.9.1 (parameters), Mastodon v4.0.0 (return value)

`Mastodon.admin_report_resolve`(*id:* AdminReport | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *AdminReport*

Marks a report as resolved (without taking any action).

The returned object reflects the updates to the report.

Added: Mastodon v2.9.1, last changed: Mastodon v2.9.1 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**admin_domain_blocks**(*id: str | int | MaybeSnowflakeIdType | datetime | None = None*, *max_id: str | int | MaybeSnowflakeIdType | datetime | None = None*, *min_id: str | int | MaybeSnowflakeIdType | datetime | None = None*, *since_id: str | int | MaybeSnowflakeIdType | datetime | None = None*, *limit: int | None = None*) → *AdminDomainBlock* | *PaginatableList*[*AdminDomainBlock*]

Fetches a list of blocked domains. Requires scope *admin:read:domain_blocks*.

Provide an *id* to fetch a specific domain block based on its database id.

Raises a *MastodonAPIError* if the specified block does not exist.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0

Mastodon.**admin_create_domain_block**(*domain: str*, *severity: str | None = None*, *reject_media: bool | None = None*, *reject_reports: bool | None = None*, *private_comment: str | None = None*, *public_comment: str | None = None*, *obfuscate: bool | None = None*) → *AdminDomainBlock*

Perform a moderation action on a domain. Requires scope *admin:write:domain_blocks*.

Valid severities are:

- "silence" - hide all posts from federated timelines and do not show notifications to local users from the remote instance's users unless they are following the remote user.
- "suspend" - deny interactions with this instance going forward. This action is reversible.
- "limit" - generally used with reject_media=true to force reject media from an instance without silencing or suspending..

If no action is specified, the domain is only silenced. *domain* is the domain to block. Note that using the top level domain will also imapct all subdomains. ie, example.com will also impact subdomain.example.com. *reject_media* will not download remote media on to your local instance media storage. *reject_reports* ignores all reports from the remote instance. *private_comment* sets a private admin comment for the domain. *public_comment* sets a publicly available comment for this domain, which will be available to local users and may be available to everyone depending on your settings. *obfuscate* censors some part of the domain name. Useful if the domain name contains unwanted words like slurs.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0 (parameters), Mastodon v4.3.0 (return value)

Mastodon.**admin_update_domain_block**(*id*, *severity: str | None = None*, *reject_media: bool | None = None*, *reject_reports: bool | None = None*, *private_comment: str | None = None*, *public_comment: str | None = None*, *obfuscate: bool | None = None*) → *AdminDomainBlock*

Modify existing moderation action on a domain. Requires scope *admin:write:domain_blocks*.

Valid severities are:

- "silence" - hide all posts from federated timelines and do not show notifications to local users from the remote instance's users unless they are following the remote user.
- "suspend" - deny interactions with this instance going forward. This action is reversible.
- "limit" - generally used with reject_media=true to force reject media from an instance without silencing or suspending.

If no action is specified, the domain is only silenced. *domain* is the domain to block. Note that using the top level domain will also imapct all subdomains. ie, example.com will also impact subdomain.example.com. *reject_media* will not download remote media on to your local instance media storage. *reject_reports* ignores all reports from the remote instance. *private_comment* sets a private admin comment for the domain. *public_comment* sets a publicly available comment for this domain, which will be available to local users and may

be available to everyone depending on your settings. *obfuscate* censors some part of the domain name. Useful if the domain name contains unwanted words like slurs.

Raises a *MastodonAPIError* if the specified block does not exist.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0 (parameters), Mastodon v4.3.0 (return value)

Mastodon.**admin_delete_domain_block**(*id=typing.Union[mastodon.return_types.AdminDomainBlock, str, int, mastodon.types_base.MaybeSnowflakeIdType, datetime.datetime]*)

Removes moderation action against a given domain. Requires scope *admin:write:domain_blocks*.

Provide an *id* to remove a specific domain block based on its database id.

Raises a *MastodonAPIError* if the specified block does not exist.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0

Mastodon.**admin_measures**(*start_at*, *end_at*, *active_users: bool = False*, *new_users: bool = False*, *interactions: bool = False*, *opened_reports: bool = False*, *resolved_reports: bool = False*, *tag_accounts:* Tag | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *tag_uses:* Tag | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *tag_servers:* Tag | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *instance_accounts: str* | *None = None*, *instance_media_attachments: str* | *None = None*, *instance_reports: str* | *None = None*, *instance_statuses: str* | *None = None*, *instance_follows: str* | *None = None*, *instance_followers: str* | *None = None*) → *NonPaginatableList*[*AdminMeasure*]

Retrieves numerical instance information for the time period (at day granularity) between *start_at* and *end_at*.

- *active_users*: Pass true to retrieve the number of active users on your instance within the time period
- *new_users*: Pass true to retrieve the number of users who joined your instance within the time period
- *interactions*: Pass true to retrieve the number of interactions (favourites, boosts, replies) on local statuses within the time period
- *opened_reports*: Pass true to retrieve the number of reports filed within the time period
- *resolved_reports* = Pass true to retrieve the number of reports resolved within the time period
- *tag_accounts*: Pass a tag ID to get the number of accounts which used that tag in at least one status within the time period
- *tag_uses*: Pass a tag ID to get the number of statuses which used that tag within the time period
- *tag_servers*: Pass a tag ID to to get the number of remote origin servers for statuses which used that tag within the time period
- *instance_accounts*: Pass a domain to get the number of accounts originating from that remote domain within the time period
- *instance_media_attachments*: Pass a domain to get the amount of space used by media attachments from that remote domain within the time period
- *instance_reports*: Pass a domain to get the number of reports filed against accounts from that remote domain within the time period
- *instance_statuses*: Pass a domain to get the number of statuses originating from that remote domain within the time period
- *instance_follows*: Pass a domain to get the number of accounts from a remote domain followed by that local user within the time period
- *instance_followers*: Pass a domain to get the number of local accounts followed by accounts from that remote domain within the time period

This API call is relatively expensive - watch your servers load if you want to get a lot of statistical data. Especially the instance_statuses stats might take a long time to compute and, in fact, time out.

There is currently no way to get tag IDs implemented in Mastodon.py, because the Mastodon public API does not implement one. This will be fixed in a future release.

Added: Mastodon v3.5.0, last changed: Mastodon v3.5.0

Mastodon.**admin_dimensions**(*start_at: datetime*, *end_at: datetime*, *limit: int | None = None*, *languages: bool = False*, *sources: bool = False*, *servers: bool = False*, *space_usage: bool = False*, *software_versions: bool = False*, *tag_servers:* Tag | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *tag_languages:* Tag | *str* | *int* | MaybeSnowflakeIdType | *datetime* | *None = None*, *instance_accounts: str | None = None*, *instance_languages: str | None = None*) → *NonPaginatableList*[*AdminDimension*]

Retrieves primarily categorical instance information for the time period (at day granularity) between *start_at* and *end_at*.

- *languages*: Pass true to get the most-used languages on this server
- *sources*: Pass true to get the most-used client apps on this server
- *servers*: Pass true to get the remote servers with the most statuses
- *space_usage*: Pass true to get the how much space is used by different components your software stack
- *software_versions*: Pass true to get the version numbers for your software stack
- *tag_servers*: Pass a tag ID to get the most-common servers for statuses including a trending tag
- *tag_languages*: Pass a tag ID to get the most-used languages for statuses including a trending tag
- *instance_accounts*: Pass a domain to get the most-followed accounts from a remote server
- *instance_languages*: Pass a domain to get the most-used languages from a remote server

Pass *limit* to set how many results you want on queries where that makes sense.

This API call is relatively expensive - watch your servers load if you want to get a lot of statistical data.

There is currently no way to get tag IDs implemented in Mastodon.py, because the Mastodon public API does not implement one. This will be fixed in a future release.

Added: Mastodon v3.5.0, last changed: Mastodon v3.5.0

Mastodon.**admin_retention**(*start_at: datetime*, *end_at: datetime*, *frequency: str = 'day'*) → *NonPaginatableList*[*AdminRetention*]

Gets user retention statistics (at *frequency* - "day" or "month" - granularity) between *start_at* and *end_at*.

Added: Mastodon v3.5.0, last changed: Mastodon v3.5.0

Mastodon.**admin_canonical_email_blocks**(*max_id: str | int |* MaybeSnowflakeIdType | *datetime | None = None*, *min_id: str | int |* MaybeSnowflakeIdType | *datetime | None = None*, *since_id: str | int |* MaybeSnowflakeIdType | *datetime | None = None*, *limit: int | None = None*) → *PaginatableList*[*AdminCanonicalEmailBlock*]

Fetches a list of canonical email blocks. Requires scope *admin:read:canonical_email_blocks*.

The returned list may be paginated using max_id, min_id, and since_id.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0

Mastodon.**admin_canonical_email_block**(*id: str | int | MaybeSnowflakeIdType | datetime*) → *AdminCanonicalEmailBlock*

Fetch a single canonical email block by ID. Requires scope *admin:read:canonical_email_blocks*.

Raises *MastodonAPIError* if the email block does not exist.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**admin_test_canonical_email_block**(*email: str*) → *NonPaginatableList*[*AdminCanonicalEmailBlock*]

Canonicalize and hash an email address, returning all matching canonical email blocks. Requires scope *admin:read:canonical_email_blocks*.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0

Mastodon.**admin_create_canonical_email_block**(*email: str | None = None*, *canonical_email_hash: str | None = None*) → *AdminCanonicalEmailBlock*

Block a canonical email. Requires scope *admin:write:canonical_email_blocks*.

Either *email* (which will be canonicalized and hashed) or *canonical_email_hash* must be provided.

Up do date details about the canonicalization and hashing process can be found here:

https://github.com/mastodon/mastodon/blob/main/app/helpers/email_helper.rb

As of Mastodon v4.4.0:

- Everything lowercased
- Dots are removed from the part before the @
- Anything after a + is removed
- The hash in use is SHA256

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**admin_delete_canonical_email_block**(*id: str | int | MaybeSnowflakeIdType | datetime*) → *AdminCanonicalEmailBlock*

Delete a canonical email block by ID. Requires scope *admin:write:canonical_email_blocks*.

Raises *MastodonAPIError* if the email block does not exist.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**admin_email_domain_blocks**(*max_id: str | int | MaybeSnowflakeIdType | datetime | None = None*, *min_id: str | int | MaybeSnowflakeIdType | datetime | None = None*, *since_id: str | int | MaybeSnowflakeIdType | datetime | None = None*, *limit: int | None = None*) → *PaginatableList*[*AdminEmailDomainBlock*]

Fetches a list of blocked email domains. Requires scope *admin:read:email_domain_blocks*.

The returned list may be paginated using max_id, min_id, and since_id.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0

Mastodon.**admin_email_domain_block**(*id: str | int | MaybeSnowflakeIdType | datetime*) → *AdminEmailDomainBlock*

Fetch a single blocked email domain by ID. Requires scope *admin:read:email_domain_blocks*.

Raises *MastodonAPIError* if the email domain block does not exist.

Added: Mastodon v4.1.0, last changed: Mastodon v4.1.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**admin_create_email_domain_block**(*domain: str*) → *AdminEmailDomainBlock*

Block an email domain from signups. Requires scope *admin:write:email_domain_blocks*.

If the domain contains invalid characters, a *MastodonAPIError* will be raised.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**admin_delete_email_domain_block**(*id: str | int | MaybeSnowflakeIdType | datetime*)

Remove an email domain block. Requires scope *admin:write:email_domain_blocks*.

Raises *MastodonAPIError* if the email domain block does not exist.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0

Mastodon.**admin_ip_blocks**(*max_id: str | int | MaybeSnowflakeIdType | datetime | None = None*, *min_id: str | int | MaybeSnowflakeIdType | datetime | None = None*, *since_id: str | int | MaybeSnowflakeIdType | datetime | None = None*, *limit: int | None = None*) → *PaginatableList*[*AdminIpBlock*]

Fetches a list of blocked IP addresses and ranges. Requires scope *admin:read:ip_blocks*.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0

Mastodon.**admin_ip_block**(*id: AdminIpBlock | str | int | MaybeSnowflakeIdType | datetime*) → *AdminIpBlock*

Fetch a single blocked IP address or range by ID. Requires scope *admin:read:ip_blocks*.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**admin_create_ip_block**(*ip: str*, *severity: str*, *comment: str | None = None*, *expires_in: int | None = None*) → *AdminIpBlock*

Block an IP address or range from signups. Requires scope *admin:write:ip_blocks*.

Provide the IP address as a CIDR range, e.g. "192.168.1.1/32" to block just that IP address, or "8.8.8.8/24" to block all addresses in the 8.8.8.* subnet.

severity can be one of three values:

- "sign_up_requires_approval" - signups from this IP will require manual approval
- "sign_up_block" - signups from this IP will be blocked
- "no_access" - all access from this IP will be blocked

expires_in is the number of seconds until the block expires. If not provided, the block will be permanent.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**admin_update_ip_block**(*id: AdminIpBlock | str | int | MaybeSnowflakeIdType | datetime*, *ip: str | None = None*, *severity: str | None = None*, *comment: str | None = None*, *expires_in: int | None = None*) → *AdminIpBlock*

Update an existing IP block. Requires scope *admin:write:ip_blocks*.

expires_in is the number of seconds until the block expires. If not provided, the block will be permanent.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0 (parameters), Mastodon v4.0.0 (return value)

Mastodon.**admin_delete_ip_block**(*id: AdminIpBlock | str | int | MaybeSnowflakeIdType | datetime*)

Remove an IP block. Requires scope *admin:write:ip_blocks*.

Added: Mastodon v4.0.0, last changed: Mastodon v4.0.0

Mastodon.**admin_trending_tags**(*limit: int | None = None*) → *NonPaginatableList*[*Tag*]

Admin version of *trending_tags()*. Includes unapproved tags.

The returned list is sorted, descending, by the instance's trending algorithm.

Returns a regular Tag without admin attributes between Mastodon.py v4.0.0 and v4.1.0 due to a bug.

Added: Mastodon v3.5.0, last changed: Mastodon v3.5.0

Mastodon.**admin_trending_statuses**() → *NonPaginatableList*[*Status*]

Admin version of *trending_statuses()*. Includes unapproved tags.

The returned list is sorted, descending, by the instance's trending algorithm.

Added: Mastodon v3.5.0, last changed: Mastodon v3.5.0

Mastodon.**admin_trending_links**() → *NonPaginatableList*[*PreviewCard*]

Admin version of *trending_links()*. Includes unapproved tags.

The returned list is sorted, descending, by the instance's trending algorithm.

Added: Mastodon v3.5.0, last changed: Mastodon v3.5.0

Mastodon.**admin_approve_trending_link**(*id:* PreviewCard | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *PreviewCard*

Approve a trending link. Requires scope *admin:write*.

Added: Mastodon v4.2.0, last changed: Mastodon v4.2.0 (parameters), Mastodon v4.3.0 (return value)

Mastodon.**admin_reject_trending_link**(*id:* PreviewCard | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *PreviewCard*

Reject a trending link. Requires scope *admin:write*.

Added: Mastodon v4.2.0, last changed: Mastodon v4.2.0 (parameters), Mastodon v4.3.0 (return value)

Mastodon.**admin_approve_trending_status**(*id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Status*

Approve a trending status. Requires scope *admin:write*.

Added: Mastodon v4.2.0, last changed: Mastodon v4.2.0 (parameters), Mastodon v4.4.0 (return value)

Mastodon.**admin_reject_trending_status**(*id:* Status | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Status*

Reject a trending status. Requires scope *admin:write*.

Added: Mastodon v4.2.0, last changed: Mastodon v4.2.0 (parameters), Mastodon v4.4.0 (return value)

Mastodon.**admin_approve_trending_tag**(*id:* Tag | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Tag*

Approve a trending tag. Requires scope *admin:write*.

Added: Mastodon v4.2.0, last changed: Mastodon v4.2.0 (parameters), Mastodon v4.4.0 (return value)

Mastodon.**admin_reject_trending_tag**(*id:* Tag | *str* | *int* | MaybeSnowflakeIdType | *datetime*) → *Tag*

Reject a trending tag. Requires scope *admin:write*.

Added: Mastodon v4.2.0, last changed: Mastodon v4.2.0 (parameters), Mastodon v4.4.0 (return value)

PYTHON MODULE INDEX

m

INDEX

Symbols

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y