aPRAW

Release 0.4.0-alpha

Jun 29, 2020

Getting Started

1 Community and Support **3**

2 Documentation Contents **5**

2.1 Introduction 5

2.2 Quickstart 5

2.3 API Reference 8

3 Index **37**

Index **39**

aPRAW is an asynchronous Reddit API wrapper.

Features:

- Modern Pythonic design using *async*/*await* syntax.
- Automatic handling of rate-limits.
- Built-in listings and streams for list endpoints.
- Compatible with previous *praw.ini* files.
- Proper OAuth2 support.

CHAPTER 1

Community and Support

If you have any questions regarding aPRAW and its usage...

- Join the /r/aPRAW subreddit
  - Feel free to post a question in the questions thread or make your own post if it could start a big discussion!
- Join the aPRAW Discord server
  - Use the `#general` chat for discussion about the library and talking to other users.
  - Use the `#questions` to post questions. The developers will try to get back to you as quickly as possible, but other users can help as well!
  - Use the `#ideas` if you have any ideas for the framework but don't know how to implement them, or just want to throw in the suggestion.

CHAPTER 2

Documentation Contents

This is the documentation for aPRAW, a wrapper library for Python to aid in performing asynchronous requests to the Reddit API and interacting with its data. It's split into the following sections.

2.1 Introduction

aPRAW serves as an asynchronous alternative to PRAW, and offers certain features and a more modern class design. Those familiar with PRAW will be able to use many features without much additional changes to the code, besides the usage of `async` and `await` syntax.

aPRAW was specifically built with Discord bots in mind, so those interested in creating a Discord bot with Discord.py and combining Reddit streams should be able to make use of its asynchronous functionalities.

2.1.1 Prerequisites

aPRAW works with Python 3.6 or higher.

2.1.2 Installing

aPRAW can be installed directly from PyPi:

```
$ pip install aPRAW
```

2.2 Quickstart

This section contains a small guide to get started with using aPRAW and its various features.

Contents

- *Quickstart*
  - *Creating a Reddit Instance*
  - *Running Asynchronous Code*
  - *Basic Concepts*
    - *Instantiating Models*
    - *Looping Through Items*
    - *Streaming Items*

2.2.1 Creating a Reddit Instance

Currently aPRAW only supports the use of a script auth flow to log in to Reddit and perform requests. Read-only modes as well as the application flow are WIP.

To obtain a `client_id` and `client_secret` for your application, head to Reddit's App Preferences and create a new app. Follow the guidelines on Reddit's Quick Start Example to obtain your credentials.

Those credentials can now be used to create a Reddit instance:

```
import apraw


# instantiate a `Reddit` instance
# you can also supply a key to an entry within a praw.ini
# file, making your login compatible with praw as well
reddit = apraw.Reddit(client_id="CLIENT_ID", client_secret="CLIENT_SECRET",
                      password="PASSWORD", user_agent="USERAGENT",
                      username="USERNAME")
```

Those previously making use of a `praw.ini` file can continue to do so, by specifying the key that was used for the client in place of the credentials. aPRAW will then automatically search for the file and save those credentials.

For more information on `praw.ini` files visit PRAW's documentation.

2.2.2 Running Asynchronous Code

Since most of aPRAW's code are asynchronous functions or generators, you will want to add your tasks to an event loop such as the `asyncio` one.

For that do the following:

```
import apraw
import asyncio

# instantiate a `Reddit` instance
reddit = apraw.Reddit(client_id="CLIENT_ID", client_secret="CLIENT_SECRET",
                      password="PASSWORD", user_agent="USERAGENT",
                      username="USERNAME")

async def scan_posts():
```

(continues on next page)

(continued from previous page)

```
  # get an instance of a subreddit
  subreddit = await reddit.subreddit("aprawtest")

  # loop through new posts
  async for submission in subreddit.new():
    print(submission.title)

if __name__ == "__main__":
    # get the asyncio event loop
    loop = asyncio.get_event_loop()

    # add scan_posts() to the queue and run it
    loop.run_until_complete(scan_posts())
```

2.2.3 Basic Concepts

aPRAW assumes that all the Reddit items know the logged-in Reddit instance. When grabbing items by using the built-in functions, this will be done automatically through dependency injection.

Instantiating Models

Most items can be retrieved from the base Reddit object like so:

```
# instantiate a `Reddit` instance
reddit = apraw.Reddit(client_id="CLIENT_ID", client_secret="CLIENT_SECRET",
                      password="PASSWORD", user_agent="USERAGENT",
                      username="USERNAME")

# grab an instance of the /r/aPRAWTest subreddit
subreddit = await reddit.subreddit("aprawtest")

# grab an instance of the /u/aPRAWBot Redditor
redditor = await reddit.redditor("aprawbot")

# grab a test submission made on /r/aPRAWTest
submission = await reddit.submission("h7mna9")

# grab a test comment made on /r/aPRAWTest
comment = await reddit.comment("fulsybg")
```

Looping Through Items

Most endpoints returning list or "*listings*" of items are represented by async generators in aPRAW. To grab a set of new posts on a subreddit try this:

```
# get an instance of a subreddit
subreddit = await reddit.subreddit("aprawtest")

# loop through new posts
async for submission in subreddit.new():
    print(submission.id)
```

In cases where *ListingGenerator* is used, `**kwargs` can be passed into the endpoint as well.

Streaming Items

ListingGenerator has a built-in *stream()* method that will poll the Reddit API endpoint it's mapped to, and yield items as they come. This is done in a very efficient manner with an internal tracker for items, an exponential function to increase wait times and the use of `asyncio.sleep()` to ensure non-blocking streams.

Polling an endpoint with *ListingGenerator* is as simple as writing:

```
# get an instance of a subreddit
subreddit = await reddit.subreddit("aprawtest")

# stream new posts
async for submission in subreddit.new.stream():
    print(submission.id)
```

2.3 API Reference

The following section outlines the API of aPRAW.

2.3.1 Reddit

User

This section describes `User` class as well as `AuthenticatedUser` that contain information about the logged-in user and request credentials.

Contents

- *User*
  - *AuthenticatedUser*
  - *Karma*

class apraw.models.**User**(*reddit: Reddit*, *username: str*, *password: str*, *client_id: str*, *client_secret: str*, *user_agent: str*)

A class to store the authentication credentials and handle ratelimit information.

reddit: Reddit The *Reddit* instance with which requests are made.

username: str The username given to the Reddit instance or obtained via `praw.ini`.

password: str The password given to the Reddit instance or obtained via `praw.ini`.

client_id: str The client ID given to the Reddit instance or obtained via `praw.ini`.

client_secret: str The client secret given to the Reddit instance or obtained via `praw.ini`.

user_agent: str The user agent given to the Reddit instance or defaulted to aPRAW's version.

password_grant: str The data to be used when making a token request with the 'password' `grant_type`.

access_data: Dict A dictionary containing the access token and user agent for request headers.

token_expires: datetime The datetime on which the previously retrieved token will expire. Defaults to the past to obtain a token immediately the first time.

ratelimit_remaining: int The number of requests remaining in the current ratelimit window.

ratelimit_used: int The number of requests previously used in the current ratelimit window.

ratelimit_reset: datetime The datetime on which the ratelimit window will be reset.

get_auth_session() → aiohttp.client.ClientSession
Retrieve an `aiohttp.ClientSesssion` with which the authentication token can be obtained.

Returns **session** – The session using the BasicAuth setup to obtain tokens with.

Return type aiohttp.ClientSession

get_client_session() → aiohttp.client.ClientSession
Retrieve the `aiohttp.ClientSesssion` with which regular requests are made.

Returns **session** – The session with which requests should be made.

Return type aiohttp.ClientSession

me() → apraw.models.user.AuthenticatedUser
Retrieve an instance of *AuthenticatedUser* for the logged-in user.

Returns **user** – The logged-in user.

Return type *AuthenticatedUser*

AuthenticatedUser

class apraw.models.**AuthenticatedUser**(*reddit: Reddit*, *data: Dict*)
The model representing the logged-in user.

This model inherits from *Redditor* and thus all its attributes and features. View those docs for further information.

reddit: Reddit The *Reddit* instance with which requests are made.

data: Dict The data obtained from the /about endpoint.

karma() → List[apraw.models.user.Karma]
Retrieve the karma breakdown for the logged-in user.

Returns **karma** – The parsed `KarmaList` for the logged-in user.

Return type List[*Karma*]

Karma

The `Karma` model represents items in a `KarmaList` and contains information about the subreddit the karma was obtained on, as well as the amount of link and comment karma.

class apraw.models.**Karma**(*reddit: Reddit*, *data: Dict*)
A model representing subreddit karma.

reddit: Reddit The *Reddit* instance with which requests are made.

data: Dict The data obtained from the /about endpoint.

Typical Attributes

This table describes attributes that typically belong to objects of this class. Attributes are dynamically provided by the *aPRAWBase* class and may vary depending on the status of the response and expected objects.

Attribute	Description
`sr`	The name of the subreddit the karma was obtained on
`comment_karma`	The amount of karma obtained on the subreddit.
`link_karma`	The amount of link karma obtained on the subreddit.

subreddit()
Retrieve the subreddit on which the karma was obtained.

Returns **subreddit** – The subreddit on which the karma was obtained.

Return type *Subreddit*

Contents

- *Reddit*

class apraw.**Reddit**(*praw_key: str = ''*, *username: str = ''*, *password: str = ''*, *client_id: str = ''*, *client_secret: str = ''*, *user_agent='aPRAW by Dan6erbond'*)
The Reddit instance with which root requests can be made.

user: User An instance of the logged-in Reddit user.

subreddits: ListingGenerator A ListingGenerator that returns newly created subreddits, which can be streamed using `reddit.subreddits.stream()`.

comment(*id: str = ''*, *url: str = ''*) → apraw.models.comment.Comment
Get a *Comment* object based on its ID or URL.

Parameters

- **id** (*str*) – The ID of a comment (with or without kind).
- **url** (*str*) – The URL of a comment.

Returns **comment** – The requested comment.

Return type *Comment*

get_listing(*endpoint: str*, *subreddit: apraw.models.subreddit.Subreddit = None*, ***kwargs*)
Retrieve a listing from an endpoint.

Parameters

- **endpoint** (*str*) – The endpoint to be appended after the base URL (https://oauth.reddit.com/).
- **subreddit** (Subreddit) – The subreddit to dependency inject into retrieved items when possible.
- **kwargs** (***Dict*) – Query parameters to be appended after the URL.

Returns **listing** – The listing containing all the endpoint's children.

Return type Listing

get_request(**args*, ***kwargs*)
Perform an HTTP GET request on the Reddit API.

Parameters

- **endpoint** (*str*) – The endpoint to be appended after the base URL (https://oauth.reddit.com/).

- **kwargs** – Query parameters to be appended after the URL.

Returns **resp** – The response JSON data.

Return type Dict or None

info(*id: str = ''*, *ids: List[str] = []*, *url: str = ''*)
Get a Reddit item based on its ID or URL.

Parameters

- **id** (*str*) – The item's ID.
- **ids** (*List[str]*) – Multiple IDs to fetch multiple items at once (max 100).
- **url** (*str*) – The item's URL.

Yields

- **comment** (*Comment*) – A *Comment* object.
- **submission** (*Submission*) – A *Submission* object.

message(*to: Union[str, apraw.models.redditor.Redditor], subject: str, text: str, from_sr: Union[str, apraw.models.subreddit.Subreddit] = ''*) → Dict
Message a Redditor or Subreddit.

Parameters

- **to** (*str or Redditor or Subreddit*) – The Redditor or Subreddit the message should be sent to.
- **subject** (*str*) – The subject of the message.
- **text** (*str*) – The text contents of the message.
- **from_sr** (*str or Subreddit*) – Optional if the message is being sent from a subreddit.

Returns **result** – The response JSON data.

Return type Dict

post_request(**args*, ***kwargs*)
Perform an HTTP POST request on the Reddit API.

Parameters

- **endpoint** (*str*) – The endpoint to be appended after the base URL (https://oauth.reddit.com/).
- **url** (*str*) – The direct URL to perform the request on.
- **data** – The data to add to the POST body.
- **kwargs** – Query parameters to be appended after the URL.

Returns **resp** – The response JSON data.

Return type Dict or None

redditor(*username: str*) → apraw.models.redditor.Redditor
Get a *Redditor* object based the Redditor's username.

Parameters **username** (*str*) – The Redditor's username (without '/u/').

Returns **redditor** – The requested Redditor, returns None if not found.

Return type *Redditor* or None

submission (*id: str = ''*, *url: str = ''*) → apraw.models.submission.Submission
Get a *Submission* object based on its ID or URL.

Parameters

- **id** (*str*) – The ID of a submission (with or without kind).
- **url** (*str*) – The URL of a submission.

Returns **submission** – The requested submission.

Return type *Submission*

subreddit (*display_name: str*) → apraw.models.subreddit.Subreddit
Get a *Subreddit* object according to the given name.

Parameters **display_name** (*str*) – The display name of the subreddit.

Returns

- **subreddit** (*Subreddit*) – The subreddit if found.
- **result** (*None*) – Returns None if subreddit not found.

2.3.2 aPRAW Models

This section contains the documentation and API of the implemented aPRAW models.

Subreddit

This section contains the documentation and API of the subreddit models and helpers.

Subreddit

This section describes the usage and members of the Subreddit model.

A subreddit can be instantiated as follows:

```
sub = await reddit.subreddit("aprawtest")
```

class apraw.models.**Subreddit** (*reddit: Reddit*, *data: Dict*)
The model representing subreddits.

reddit: Reddit The *Reddit* instance with which requests are made.

data: Dict The data obtained from the /about endpoint.

kind: str The item's kind / type.

mod: SubredditModeration Returns an instance of *SubredditModeration*.

modmail: SubredditModmail Returns an instance of *SubredditModmail*.

wiki: SubredditWiki Returns an instance of SubredditWiki.

comments: ListingGenerator Returns an instance of *ListingGenerator* mapped to the comments endpoint.

new: ListingGenerator Returns an instance of *ListingGenerator* mapped to the new submissions endpoint.

hot: ListingGenerator Returns an instance of *ListingGenerator* mapped to the hot submissions endpoint.

rising: ListingGenerator Returns an instance of *ListingGenerator* mapped to the rising submissions endpoint.

top: ListingGenerator Returns an instance of *ListingGenerator* mapped to the top submissions endpoint.

Warning: Using the streams of non-new endpoints may result in receiving items multiple times, as their positions can change and be returned by the API after they've been removed from the internal tracker.

Examples

To grab new submissions made on a subreddit:

```
sub = await reddit.subreddit("aprawtest")
async for submission in sub.new(): # use .new.stream() for endless polling
    print(submission.title, submission.body)
```

Typical Attributes

This table describes attributes that typically belong to objects of this class. Attributes are dynamically provided by the *aPRAWBase* class and may vary depending on the status of the response and expected objects.

Attribute	Description
accounts_active_is_fuzzed	bool
accounts_active	null
active_user_count	The number of active users on the subreddit.
advertiser_category	string
all_original_content	Whether the subreddit requires all content to be OC.
allow_discovery	Whether the subreddit can be discovered.
allow_images	Whether images are allowed as submissions.
allow_videogifs	Whether GIFs are allowed as submissions.
allow_videos	Whether videos are allowed as submissions.
banner_background_color	The banner's background color if applicable, otherwise empty.
banner_background_image	A URL to the subreddit's banner image.
banner_img	A URL to the subreddit's banner image if applicable.
banner_size	The subreddit's banner size if applicable.
can_assign_link_flair	Whether submission flairs can be assigned.
can_assign_user_flair	Whether the user can assign their own flair on the subreddit.
collapse_deleted_comments	Whether deleted comments should be deleted by clients.
comment_score_hide_mins	The minimum comment score to hide.
community_icon	A URL to the subreddit's community icon if applicable.
created_utc	The date on which the subreddit was created in UTC datetime.
created	The time the subreddit was created on.
description_html	The subreddit's description as HTML.
description	The subreddit's short description.
disable_contributor_requests	bool
display_name_prefixed	The subreddit's display name prefixed with 'r/'.
display_name	The subreddit's display name.
emojis_custom_size	The custom size set for emojis.
emojis_enabled	Whether emojis are enabled on this subreddit.

Continued on next page

Table 1 – continued from previous page

Attribute	Description
free_form_reports	Whether it's possible to submit free form reports.
has_menu_widget	Whether the subreddit has menu widgets.
header_img	A URL to the subreddit's header image of applicable.
header_size	The subreddit's header size.
header_title	The subreddit's header title.
hide_ads	Whether ads are hidden on this subreddit.
icon_img	A URL to the subreddit's icon image of applicable.
icon_size	The subreddit's icon size.
id	The subreddit's ID.
is_enroled_in_new_modmail	Whether the subreddit is enrolled in new modmail.
key_color	string
lang	The subreddit's language.
link_flair_enabled	Whether link flairs have been enabled for the subreddit.
link_flair_position	The position of link flairs.
mobile_banner_size	A URL to the subreddit's mobile banner if applicable.
name	The subreddit's fullname (t5_ID).
notification_level	
original_content_tag_enabled	Whether the subreddit has the OC tag enabled.
over18	Whether the subreddit is NSFW.
primary_color	The subreddit's primary color.
public_description_html	The subreddit's public description as HTML.
public_description	The subreddit's public description string.
public_traffic	bool
quarantine	Whether the subreddit is quarantined.
restrict_commenting	Whether comments by users are restricted on the subreddit.
restrict_posting	Whether posts to the subreddit are restricted.
show_media_preview	Whether media previews should be displayed by clients.
show_media	
spoilers_enabled	Whether the spoiler tag is enabled on the subreddit.
submission_type	The types of allowed submissions. Default is "any".
submit_link_label	The subreddit's submit label if applicable.
submit_text_html	The HTML submit text if a custom one is set on the subreddit.
submit_text_label	The text used for the submit button.
submit_text	The markdown submit text if a custom one is set on the subreddit.
subreddit_type	The subreddit type, either "public", "restricted" or "private".
subscribers	The number of subreddit subscribers.
suggested_comment_sort	The suggested comment sort algorithm, can be null.
title	The subreddit's banner title.
url	The subreddit's display name prepended with "/r/".
user_can_flair_in_sr	Whether the user can assign custom flairs (nullable).
user_flair_background_color	The logged in user's flair background color if applicable.
user_flair_css_class	The logged in user's flair CSS class.
user_flair_enabled_in_sr	Whether the logged in user's subreddit flair is enabled.
user_flair_position	The position of user flairs on the subreddit (right or left).
user_flair_richtext	The logged in user's flair text if applicable.
user_flair_template_id	The logged in user's flair template ID if applicable.
user_flair_text_color	The logged in user's flair text color.
user_flair_text	The logged in user's flair text.
user_flair_type	The logged in user's flair type.

Continued on next page

Table 1 – continued from previous page

Attribute	Description
user_has_favorited	Whether the logged in user has favorited the subreddit.
user_is_banned	Whether the logged in user is banned from the subreddit.
user_is_contributor	Whether the logged in user has contributed to the subreddit.
user_is_moderator	Whether the logged in user is a moderator on the subreddit.
user_is_muted	Whether the logged in user has been muted by the subreddit.
user_is_subscriber	Whether the logged in user is subscribed to the subreddit.
user_sr_flair_enabled	Whether the logged in user's subreddit flair is enabled.
user_sr_theme_enabled	Whether the logged in user has enabled the custom subreddit theme.
videostream_links_count	The number of submissions with videostream links.
whitelist_status	
wiki_enabled	Whether the subreddit has the wiki enabled.
wls	null

message (*subject: str*, *text: str*, *from_sr: Union[str*, *Subreddit]* = '') → Dict
Send a message to the subreddit.

Parameters

- **subject** (*str*) – The message subject.
- **text** (*str*) – The message contents as markdown.
- **from_sr** (*str or Subreddit*) – The subreddit the message is being sent from if applicable.

Returns **response** – The API response JSON as a dictionary.

Return type Dict

moderators (**kwargs) → AsyncIterator[apraw.models.subreddit.SubredditModerator]
Yields all the subreddit's moderators.

Parameters **kwargs** (***Dict*) – The query parameters to be added to the GET request.

Yields **moderator** (*SubredditModerator*) – An instance of the moderators as *SubredditModerator*.

Subreddit Moderation

This section details the usage of models related to subreddit moderation.

Contents

- *Subreddit Moderation*
  - *SubredditModerator*
  - *SubredditModeration*
  - *ModAction*

SubredditModerator

Subreddit moderators are usually retrieved as follows:

```
sub = await reddit.subreddit("aprawtest")
moderators = []
async for moderator in sub.moderators():
    moderators.append(str(moderator))
```

class apraw.models.**SubredditModerator**(*reddit: Reddit*, *data: Dict*)

The model representing subreddit moderators. Redditors can be retrieved via `redditor()`.

Typical Attributes

This table describes attributes that typically belong to objects of this class. Attributes are dynamically provided by the *aPRAWBase* class and may vary depending on the status of the response and expected objects.

Attribute	Description
`added`	The date on which the moderator was added.
`author_flair_css_class`	The moderator's flair CSS class in the respective subreddit.
`author_flair_text`	The moderator's flair text in the respective subreddit.
`id`	The Redditor's fullname (t2_ID).
`mod_permissions`	A list of all the moderator permissions or `["all"]`.
`name`	The Redditor's name.

redditor() → apraw.models.redditor.Redditor

Retrieve the Redditor this Moderator represents.

Returns **redditor** – The Redditor that is represented by this object.

Return type *Redditor*

SubredditModeration

Items in the modqueue can be fetched using the `modqueue` listing:

```
sub = await reddit.subreddit("aprawtest")
async for item in sub.mod.modqueue(): # can also be streamed
    print(type(item))
    >>> apraw.models.Comment or apraw.models.Submission
```

class apraw.models.**SubredditModeration**(*subreddit*)

A helper class for grabbing listings to Subreddit moderation items.

reports: ListingGenerator Returns an instance of *ListingGenerator* mapped to grab reported items.

spam: ListingGenerator Returns an instance of *ListingGenerator* mapped to grab items marked as spam.

modqueue: ListingGenerator Returns an instance of *ListingGenerator* mapped to grab items in the modqueue.

unmoderated: ListingGenerator Returns an instance of *ListingGenerator* mapped to grab unmoderated items.

edited: ListingGenerator Returns an instance of *ListingGenerator* mapped to grab edited items.

log: ListingGenerator Returns an instance of *ListingGenerator* mapped to grab mod actions in the subreddit log.

ModAction

class apraw.models.**ModAction**(*reddit*, *data*, *subreddit=None*)

A model representing mod actions taken on specific items.

reddit: Reddit The *Reddit* instance with which requests are made.

data: Dict The data obtained from the /about endpoint.

kind: str The item's kind / type.

Typical Attributes

This table describes attributes that typically belong to objects of this class. Attributes are dynamically provided by the *aPRAWBase* class and may vary depending on the status of the response and expected objects.

Attribute	Description
action	The type of action performed.
created_utc	The parsed UTC datetime of when the action was performed.
description	The description added to the action if applicable.
details	The details of the action performed.
id	The ID of the mod action prepended with "*ModAction*".
mod_id36	The ID36 of the moderator who performed the action.
mod	The username of the moderator who performed the action.
sr_id36	The ID36 of the subreddit the action was performed on.
subreddit_name_prefixed	The name of the subreddit the action was performed on prefixed with "r/".
subreddit	The name of the subreddit the action was performed on.
target_author	The author of the target item if applicable.
target_body	The body of the target item if applicable.
target_fullname	The id of the target with its kind prepended. (e.g. "t3_d5229o")
target_permalink	The target of the comment or submission if applicable.
target_title	The title of the submission if applicable.

mod() → apraw.models.redditor.Redditor

Returns the Redditor who performed this action.

Returns **redditor** – The Redditor who performed this action.

Return type *Redditor*

Subreddit Modmail

This section details the usage of models related to subreddit modmail.

Contents

- *Subreddit Modmail*
  - *ModmailMessage*
  - *SubredditModmail*
  - *ModmailConversation*

ModmailMessage

class apraw.models.**ModmailMessage**(*conversation: apraw.models.modmail.ModmailConversation*, *data: Dict*)

The model for modmail messages.

conversation: ModmailConversation The *ModmailConversation* instance this message belongs to.

data: Dict The data obtained from the API.

id: str The ID of this message.

body: str The HTML body of this message.

body_md: str The raw body of this message.

is_internal: str Whether the message was sent internally.

date: str A timestamp on which the message was sent.

Note: ModmailMessage attributes are loaded statically, meaning they will always be present under the abovementioned names.

---

author() → Redditor

Retrieve the author of this message as a *Redditor*.

Returns **author** – The author of this modmail message.

Return type *Redditor*

SubredditModmail

class apraw.models.**SubredditModmail**(*subreddit: Subreddit*)

Helper class to aid in retrieving subreddit modmail.

subreddit: Subreddit The subreddit this helper operates under.

conversations() → apraw.models.modmail.ModmailConversation

Retrieve a list of modmail conversations.

Yields **conversation** (*ModmailConversation*) – A modmail conversation held in the subreddit.

ModmailConversation

class apraw.models.**ModmailConversation**(*reddit: Reddit*, *data: Dict*, *owner: Subreddit = None*)

The model for modmail conversations.

reddit: Reddit The *Reddit* instance with which requests are made.

data: Dict The data obtained from the /about endpoint.

Typical Attributes

This table describes attributes that typically belong to objects of this class. Attributes are dynamically provided by the *aPRAWBase* class and may vary depending on the status of the response and expected objects.

Attribute	Description
`isAuto`	`bool`
`objIds`	A list of dictionaries containing the objects with their IDs and keys.
`isRepliable`	Whether the conversation can be replied to.
`lastUserUpdate`	A timestamp of the last user update or `None`.
`isInternal`	Whether it's an internal mod conversation.
`lastModUpdate`	A timestamp of the last moderator update or `None`.
`lastUpdated`	A timestamp of the last update made overall.
`authors`	A list of dictionaries containing authors by name with additional meta information such as `isMod`, `isAdmin`, `isOp`, `isParticipant`, `isHidden`, `id`, `isDeleted`.
`owner`	A dictionary describing the subreddit this conversation is held in.
`id`	The ID of this conversation.
`isHighlighted`	Whether the conversation has been highlighted.
`subject`	The subject of this conversation.
`participant`	`Dict`
`state`	`int`
`lastUnread`	`None`
`numMessages`	The number of messages in this conversation.

full_data() → Dict
Retrieve the raw full data from the `/api/mod/conversations/{id}` endpoint.

Returns **full_data** – The full data retrieved from the endpoint.

Return type Dict

messages() → apraw.models.modmail.ModmailMessage
Retrieve the messages sent in this conversation.

Yields **message** (*ModmailMessage*) – A message sent in this conversation.

owner() → Subreddit
Retrieve the owner subreddit of this conversation.

Returns **owner** – The subreddit this conversation was held in.

Return type *Subreddit*

Submission

This section contains the documentation and API of the submission model and its moderation helper class.

Submission

A Submission can either be instantiated by using its ID, or by going through subreddits:

```
submission = await reddit.submission("h7mna9")

sub = await reddit.redditor("aprawbot")
async for submission in sub.new():
    print(submission)
```

class apraw.models.**Submission**(*reddit: Reddit*, *data: Dict*, *full_data: Dict = None*, *subreddit: apraw.models.subreddit.Subreddit = None*, *author: apraw.models.redditor.Redditor = None*)

The model representing submissions.

reddit: Reddit The *Reddit* instance with which requests are made.

data: Dict The data obtained from the /about endpoint.

mod: SubmissionModeration The *SubmissionModeration* instance to aid in moderating the submission.

kind: str The item's kind / type.

Typical Attributes

This table describes attributes that typically belong to objects of this class. Attributes are dynamically provided by the *aPRAWBase* class and may vary depending on the status of the response and expected objects.

Attribute	Description
all_awardings	A list of the awardings on the submission.
allow_live_comments	Whether live comments have been enabled on this submission.
approved_at_utc	The UTC timestamp of when the submission was approved.
approved_by	The user that approved the submission.
approved	Whether the submission has been approved by the moderators of the subreddit.
archived	Whether the submission has been archived by Reddit.
author_flair_background_color	The submission author's flair background color.
author_flair_css_class	The submission's author flair CSS class.
author_flair_richtext	The submission's author flair text.
author_flair_template_id	The submission author's flair template ID if applicable.
author_flair_text_color	The submission's author flair text color if applicable.
author_flair_text	The author's flair text if applicable.
author_flair_type	The type of flair used by the submission's author.
author_fullname	The author of the submission prepended with t2_.
author_patreon_flair	The submission's author Patreon flair.
author	The name of the submission's Redditor.
banned_at_utc	The UTC timestamp at which the author was banned.
banned_by	null
can_gild	Whether the logged-in user can gild the submission.
can_mod_post	Whether the logged-in user can modify the post.
category	The submission's category.
clicked	Whether the submission has been clicked by the logged-in user previously.
content_categories	The content categories assigned to the submission.
contest_mode	Whether the moderators of the subreddit have enabled contest mode on the submission.
created_utc	The parsed UTC datetime on which the submission was made.
created	The timestamp of when the submission was posted.
discussion_type	null
distinguished	The type of distinguishment on the submission.
domain	The domain of the submission.
downs	The number of downvotes on the submission.
edited	Whether the submission has been edited by its author.
gilded	The number of awards this submission has received.
gildings	The gild awards the submission has received.
hidden	Whether the submission has been hidden by the logged-in user.
hide_score	Whether clients should hide the score from users.

Continued on next page

Table 2 – continued from previous page

Attribute	Description
id	The submission's ID.
ignore_reports	Whether reports should be ignored on this submission.‘‘
is_crosspostable	Whether the submission can be crossposted to other subreddits.
is_meta	Whether the submission is a meta post.
is_original_content	Whether the submission has been marked as original content.
is_reddit_media_domain	Whether the media has been uploaded to Reddit.
is_robot_indexable	Whether the submission can be indexed by robots.
is_self	Whether the submission is a self post.
is_video	Whether the submission is a video post.
likes	bool
link_flair_background_color	The submission's flair background color.
link_flair_css_class	The CSS class applied on the submission's flair if applicable.
link_flair_richtext	The submission's flair text if applicable.
link_flair_template_id	The submission's flair template ID if applicable.
link_flair_text_color	The submission's flair text color if applicable.
link_flair_text	The submission's flair text.
link_flair_type	The type of flair applied to the submission.
locked	Whether the submission has been locked by the subreddit moderators.
media_embed	Dict
media_only	Whether the submission only consists of media.
media	null
mod_note	Moderator notes added to the submission.
mod_reason_by	The moderator who added the removal reason if applicable.
mod_reason_title	The reason the submission has been removed by moderators if applicable.
mod_reports	A list of moderator reports on the submission.
name	The ID of the submission prepended with t3_.
no_follow	bool
num_comments	The number of comments on the submission.
num_crossposts	The number of times the submission has been crossposted.
num_reports	The number of reports on the submission.
over_18	Whether the submission has been marked as NSFW.
parent_whitelist_status	null
permalink	The submission's permalink.
pinned	Whether the submission has been pinned on the subreddit.
pwls	null
quarantine	Whether the submission was posted in a quarantined subreddit.
removal_reason	The submission's removal reason if applicable.
removed	Whether the submission has been removed by the subreddit moderators.
report_reasons	A list of report reasons on the submission.
saved	Whether the submission has been saved by the logged-in user.
score	The overall submission vote score.
secure_media_embed	Dict
secure_media	null
selftext_html	The submission text as HTML.
selftext	The submission's selftext.
send_replies	Whether the author of the submission will receive reply notifications.
spam	Whether the submission has been marked as spam.
spoiler	Whether the submission contains a spoiler.
stickied	Whether the submission is stickied on the subreddit.

Continued on next page

Table 2 – continued from previous page

Attribute	Description
subreddit_id	The subreddit's ID prepended with t5_.
subreddit_name_prefixed	The name of the subreddit the submission was posted on, prefixed with "r/".
subreddit_subscribers	The number of subscribers to the submission's subreddit.
subreddit_type	The type of the subreddit the submission was posted on (public, restricted, private).
subreddit	The name of the subreddit on which the submission was posted.
suggested_sort	The suggested sort method for comments.
thumbnail_height	The height of the submission's thumbnail if applicable.
thumbnail_width	The width of the submission's thumbnail if applicable.
thumbnail	A URL to the submission's thumbnail if applicable.
title	The submission's title.
total_awards_received	The number of awards on the submission.
ups	The number of upvotes on the submission.
url	The full URL of the submission.
user_reports	A list of the user reports on the submission.
view_count	The number of views on the submission.
visited	Whether the logged-in user has visited the submission previously.
whitelist_status	null
wls	null

Note: Many of these attributes are only available if the logged-in user has moderator access to the item.

author() → apraw.models.redditor.Redditor
Retrieve the item's author as a *Redditor*.

Returns **author** – The item's author.

Return type *Redditor*

clear_vote()
Clear user up- and downvotes on the item.

Returns **resp** – The API response JSON.

Return type Dict

comments(*reload=False*, ***kwargs*) → AsyncIterator[apraw.models.comment.Comment]
Iterate through all the comments made in the submission.

This endpoint retrieves all comments found in the full data retrieved from the /r/{sub}/comments/{id} endpoint, as well as /api/morechildren. *morechildren()* usually won't need to be called by end users of aPRAW.

Parameters

- **reload** (*bool*) – Whether to force reload the data.

  Warning: reload and refresh arguments will be replaced by refreshables in future releases of aPRAW, as they are alpha features.

- **kwargs** (***Dict*) – Query parameters to append to the request URL.

Yields **comment** (*Comment*) – A comment made in the submission.

delete()
Delete the item.

Returns **resp** – The API response JSON.

Return type Dict

downvote()
Downvote the item.

Returns **resp** – The API response JSON.

Return type Dict

full_data() → Dict
Retrieve the submission's full data from the /r/{sub}/comments/{id} endpoint.

Returns **full_data** – The full data retrieved from the /r/{sub}/comments/{id} endpoint.

Return type Dict

hide()
Hide the item.

Returns **resp** – The API response JSON.

Return type Dict

mark_nsfw()
Mark the item as NSFW.

Returns **resp** – The API response JSON.

Return type Dict

morechildren(*children*) → List[apraw.models.comment.Comment]
Retrieves further comments made in the submission.

Parameters **children** (*List[str]*) – A list of comment IDs to retrieve.

Returns **comments** – A list of the comments retrieved from the endpoint using their IDs.

Return type List[*Comment*]

save(*category: str = ''*)
Save the item in a category.

Parameters **category** (*str, optional*) – The category name.

Returns **resp** – The API response JSON.

Return type Dict

spoiler()
Mark the item as a spoiler.

Returns **resp** – The API response JSON.

Return type Dict

subreddit() → apraw.models.subreddit.Subreddit
Retrieve the subreddit this item was made in as a *Subreddit*.

Returns **subreddit** – The subreddit this item was made in.

Return type *Subreddit*

unhide()
Unhide the item.

Returns **resp** – The API response JSON.

Return type Dict

unmark_nsfw()
Unmark the item as NSFW.

Returns **resp** – The API response JSON.

Return type Dict

unsave()
Unsave the item.

Returns **resp** – The API response JSON.

Return type Dict

unspoiler()
Unmark the item as a spoiler.

Returns **resp** – The API response JSON.

Return type Dict

upvote()
Upvote the item.

Returns **resp** – The API response JSON.

Return type Dict

Submission Moderation

class apraw.models.**SubmissionModeration**(*reddit: Reddit*, *submission: apraw.models.submission.Submission*)
A helper class to moderate submissions.

reddit: Reddit The *Reddit* instance with which requests are made.

fullname: str The ID prepended with the kind of the item this helper belongs to.

approve()
Approve the Reddit item.

Returns **resp** – The API response JSON.

Return type Dict

distinguish(*how: NewType.<locals>.new_type = 'yes'*, *sticky: bool = False*)
Distinguish the Reddit item.

Parameters

- **how** (*"yes" or "no" or "admin" or "special"*) – The type of distinguishment to be added to the item.
- **sticky** (*bool, optional*) – Whether the item should be stickied.

Returns **resp** – The API response JSON.

Return type Dict

ignore_reports()
Ignore reports on the Reddit item.

Returns **resp** – The API response JSON.

Return type Dict

lock()
Lock the item from further replies.

Returns **resp** – The API response JSON.

Return type Dict

mark_nsfw()
Mark the item as NSFW.

Returns **resp** – The API response JSON.

Return type Dict

remove()
Remove the Reddit item.

Returns **resp** – The API response JSON.

Return type Dict

spoiler()
Mark the item as a spoiler.

Returns **resp** – The API response JSON.

Return type Dict

sticky(*position: int = 1*, *to_profile: bool = False*)
Sticky a submission in its subreddit.

Parameters

- **position** (*int*) – The "slot" the submission will be stickied to.
- **to_profile** (*bool*) – Whether the submission will be stickied to the user profile.

Returns **resp** – The API response JSON.

Return type Dict

undistinguish()
Undistinguish the Reddit item.

Returns **resp** – The API response JSON.

Return type Dict

unignore_reports()
Unignore previously ignored reports on the Reddit item.

Returns **resp** – The API response JSON.

Return type Dict

unlock()
Unlock the item from further replies.

Returns **resp** – The API response JSON.

Return type Dict

unmark_nsfw()
Unmark the item as NSFW.

Returns **resp** – The API response JSON.

Return type Dict

unspoiler()
Unmark the item as a spoiler.

Returns **resp** – The API response JSON.

Return type Dict

unsticky(*to_profile: bool = False*)
Unsticky a submission from its subreddit.

Parameters **to_profile** (*bool*) – Whether the submission will be unstickied from the user profile.

Returns **resp** – The API response JSON.

Return type Dict

Comment

This section contains the documentation and API of the comment model and its moderation helper class.

Comment

Besides retrieving comments similarly to submissions using their ID or fetching them through a subreddit's listings, comments can be obtained from the submission they were made in like so:

```
submission = await reddit.submission("h7mna9")

async for comment in submission.comments():
    print(comment)
```

class apraw.models.**Comment**(*reddit: Reddit, data: Dict, submission: Submission = None, author: apraw.models.redditor.Redditor = None, subreddit: apraw.models.subreddit.Subreddit = None, replies: List[Comment] = None*)
The model representing comments.

reddit: Reddit The *Reddit* instance with which requests are made.

data: Dict The data obtained from the /about endpoint.

mod: CommentModeration The *CommentModeration* instance to aid in moderating the comment.

kind: str The item's kind / type.

url: str The URL pointing to this comment.

Typical Attributes

This table describes attributes that typically belong to objects of this class. Attributes are dynamically provided by the *aPRAWBase* class and may vary depending on the status of the response and expected objects.

Attribute	Description
all_awardings	A list of awardings added to the comment.
approved_at_utc	The UTC timestamp at which the comment was approved by the moderators.
approved_by	The moderator who approved this comment if applicable.
approved	Whether the comment has been approved by the moderators.
archived	Whether the comment has been archived.
author_flair_background_color	The comment author's flair background color if applicable.
author_flair_css_class	The comment author's flair CSS class if applicable.
author_flair_richtext	The comment author's flair text if applicable.
author_flair_template_id	The comment author's flair template ID if applicable.
author_flair_text_color	The comment author's flair text color if applicable.
author_flair_text	The comment author's flair text if applicable.
author_flair_type	The comment author's flair type if applicable.
author_fullname	The comment author's ID prepended with t2_.
author_patreon_flair	The comment author's Patreon flair if applicable.
author	The comment author's username.
banned_at_utc	None
banned_by	None
body_html	The HTML version of the comment's body.
body	The comment's markdown body.
can_gild	Whether the logged-in user can gild the comment.
can_mod_post	bool
collapsed_reason	None
collapsed	Whether the comment should be collapsed by clients.
controversiality	A score on the comment's controversiality based on its up- and downvotes.
created_utc	The parsed UTC datetime on which the comment was made.
created	A timestamp on which the comment was created.
distinguished	The type of distinguishment the comment hsa received.
downs	The number of downvotes the comment has received.
edited	Whether the comment has been edited from its original state.
gilded	The number of awards this comment has received.
gildings	A dictionary of gilds the comment has received.
id	The comment's ID.
ignore_reports	Whether reports should be ignored on this comment.
is_submitter	Whether the logged-in user is the submitter of this comment.
likes	The overall upvote score on this comment.
link_author	The username of the comment submission's author.
link_id	The ID of the submission this comment was made in.
link_permalink / link_url	A URL to the comment's submission.
link_title	The comment's submission title.
locked	Whether the comment has been locked by the moderators.
mod_note	Notes added to the comment by moderators if applicable.
mod_reason_by	The moderator who added a removal reason if applicable.
mod_reason_title	The mod reason's title if applicable.
mod_reports	A list of reports made on this comment filed by moderators.
name	The comment's ID prepended with t1_.
no_follow	bool
num_comments	The number of replies made in this submission.
num_reports	The number of reports on this comment.
over_18	Whether the comment has been marked NSFW.
parent_id	The comment's parent ID, either link_id or the ID of another comment.

Continued on next page

Table 3 – continued from previous page

Attribute	Description
permalink	The comment's permalink.
quarantine	bool
removal_reason	A removal reason set by moderators if applicable.
removed	Whether the comment has been removed by the moderators of the subreddit.
replies	A list of replies made under this comment, usually empty at first.
report_reasons	Report reasons added to the comment.
saved	Whether the logged-in user has saved this comment.
score_hidden	Whether clients should hide the comment's score.
score	The overall upvote score on this comment.
send_replies	Whether the OP has enabled reply notifications.
spam	Whether the comment has been flagged as spam.
stickied	Whether the comment has been stickied by the moderators.
subreddit_id	The comment subreddit's ID prepended with t5_.
subreddit_name_prefixed	The comment's subreddit name prefixed with "r/".
subreddit_type	The type of the subreddit the submission was posted on (public, restricted, private).
subreddit	The name of the subreddit this comment was made in.
total_awards_received	The number of awards this comment has received.
ups	The number of upvotes this comment has received.
user_reports	A list of user reports filed for this comment.

Note: Many of these attributes are only available if the logged-in user has moderator access to the item.

author() → apraw.models.redditor.Redditor
Retrieve the item's author as a *Redditor*.

Returns **author** – The item's author.

Return type *Redditor*

clear_vote()
Clear user up- and downvotes on the item.

Returns **resp** – The API response JSON.

Return type Dict

delete()
Delete the item.

Returns **resp** – The API response JSON.

Return type Dict

downvote()
Downvote the item.

Returns **resp** – The API response JSON.

Return type Dict

full_data(*refresh: bool = False*) → Dict
Retrieve the submission's full data from the /r/{sub}/comments/{submission}/_/{id} endpoint.

Returns **full_data** – The full data retrieved from the API.

Return type Dict

hide()
Hide the item.

Returns **resp** – The API response JSON.

Return type Dict

refresh()
Reload the comment's data and replies.

> **Warning:** Refresh methods will be replaced by refreshables in future releases of aPRAW, and these methods will not be available in post-alpha releases.

replies(*refresh: bool = False*) → AsyncIterator[apraw.models.comment.Comment]
Retrieve this comment's replies.

> **Note:** Replies are returned as *Comment* and already have their `_replies` recursively filled with data retrieved from the request made originally. Fetching replies at a further depth will not result in further calls unless specifically specified with the `refresh` argument.

Parameters **refresh** (*bool*) – Whether to force a refresh of previously fetched comments.

> **Warning:** `reload` and `refresh` arguments will be replaced by refreshables in future releases of aPRAW, as they are alpha features.

Yields **reply** (*Comment*) – A reply to this comment.

save(*category: str = ''*)
Save the item in a category.

Parameters **category** (*str, optional*) – The category name.

Returns **resp** – The API response JSON.

Return type Dict

submission() → Submission
Retrieve the submission this comment was made in as a *Submission*.

Returns **submission** – The submission this comment was made in.

Return type *Submission*

subreddit() → apraw.models.subreddit.Subreddit
Retrieve the subreddit this item was made in as a *Subreddit*.

Returns **subreddit** – The subreddit this item was made in.

Return type *Subreddit*

unhide()
Unhide the item.

Returns **resp** – The API response JSON.

Return type Dict

unsave()
Unsave the item.

Returns **resp** – The API response JSON.

Return type Dict

upvote()
Upvote the item.

Returns **resp** – The API response JSON.

Return type Dict

Comment Moderation

class apraw.models.**CommentModeration**(*reddit: Reddit*, *comment: apraw.models.comment.Comment*)
A helper class to moderate comments.

reddit: Reddit The *Reddit* instance with which requests are made.

fullname: str The ID prepended with the kind of the item this helper belongs to.

approve()
Approve the Reddit item.

Returns **resp** – The API response JSON.

Return type Dict

distinguish(*how: NewType.<locals>.new_type = 'yes'*, *sticky: bool = False*)
Distinguish the Reddit item.

Parameters

- **how** (*"yes" or "no" or "admin" or "special"*) – The type of distinguishment to be added to the item.
- **sticky** (*bool, optional*) – Whether the item should be stickied.

Returns **resp** – The API response JSON.

Return type Dict

ignore_reports()
Ignore reports on the Reddit item.

Returns **resp** – The API response JSON.

Return type Dict

lock()
Lock the item from further replies.

Returns **resp** – The API response JSON.

Return type Dict

remove()
Remove the Reddit item.

Returns **resp** – The API response JSON.

Return type Dict

show_comment()
Mark a comment that it should not be collapsed because of crowd control.

The comment could still be collapsed for other reasons.

Returns **resp** – The API response JSON.

Return type Dict

undistinguish()
Undistinguish the Reddit item.

Returns **resp** – The API response JSON.

Return type Dict

unignore_reports()
Unignore previously ignored reports on the Reddit item.

Returns **resp** – The API response JSON.

Return type Dict

unlock()
Unlock the item from further replies.

Returns **resp** – The API response JSON.

Return type Dict

Redditor

This section describes the usage and members of the Redditor model.

A Redditor can be instantiated as follows:

```
sub = await reddit.redditor("aprawbot")
```

class apraw.models.**Redditor**(*reddit: Reddit*, *data: Dict*)
The model representing Redditors.

reddit: Reddit The *Reddit* instance with which requests are made.

data: Dict The data obtained from the /about endpoint.

kind: str The item's kind / type.

comments: ListingGenerator Returns an instance of *ListingGenerator* mapped to fetch the Redditor's comments.

submissions: ListingGenerator Returns an instance of *ListingGenerator* mapped to fetch the Redditor's submission.

subreddit: Sureddit An instance of *Subreddit* for the Redditor's profile subreddit.

Typical Attributes

This table describes attributes that typically belong to objects of this class. Attributes are dynamically provided by the *aPRAWBase* class and may vary depending on the status of the response and expected objects.

Attribute	Description
comment_karma	The amount of comment karma the Redditor has obtained.
created_utc	The date on which the Redditor was created in UTC datetime.
created	The timestamp of when the Redditor was created.
has_verified_email	Whether the Redditor has a verified email address.
icon_img	A URL to the Redditor's icon image if applicable.
id	The Redditor's ID (without kind).
is_employee	Whether the Redditor is a Reddit employee.
is_friend	Whether the Redditor has been added as a friend.
is_gold	Whether the Redditor is a Reddit gold member.
is_mod	Whether the Redditor is a moderator in a subreddit.
is_suspended	Whether the Redditor has been suspended.
link_karma	The amount of link karma the Redditor has obtained.
name	The Redditor's username.
pref_show_snoovatar	Whether to show the Redditor's Snoovatar in place of their icon.
verified	Whether the Redditor is verified.

Warning: Suspended Redditors only return `is_suspended` and `name`.

message (*subject*, *text*, *from_sr=''*) → Dict
Message the Redditor.

Parameters

- **subject** (*str*) – The subject of the message.
- **text** (*str*) – The text contents of the message in markdown.
- **from_sr** (*str*) – The subreddit the message is being sent from if applicable.

Returns **resp** – The response data returned from the endpoint.

Return type Dict

moderated_subreddits (**kwargs*) → Subreddit
Yields the subreddits the Redditor moderates.

Parameters **kwargs** (***Dict*) – `kwargs` to be used as query parameters.

Yields **subreddit** (*Subreddit*) – A subreddit the user moderates.

2.3.3 Helpers

This section contains the documentation of implemented base and helper classes used by aPRAW models.

ListingGenerator

`ListingGenerator` is a utility class that fetches items from the listing endpoint, parses the response, and yields items as they are found. If the item kind cannot be identified, *aPRAWBase* is returned which automatically assigns itself all the data attributes found.

class apraw.models.**ListingGenerator**(*reddit: Reddit*, *endpoint: str*, *max_wait: int = 16*, *kind_filter: List[str] = []*, *subreddit=None*)
The model to request, parse and poll listings from Reddit.

reddit: Reddit The *Reddit* instance with which requests are made.

endpoint: str The endpoint to make requests on.

max_wait: int The maximum amount of seconds to wait before re-requesting in streams.

kind_filter: Kinds to return if given, otherwise all are returned.

subreddit: Subreddit The subreddit to inject as a dependency into items if given.

Note: ListingGenerator will automatically make requests until none more are found or the limit has been reached.

---

get (*limit: int = 25*, ***kwargs*) → AsyncIterator[apraw.models.helpers.apraw_base.aPRAWBase]
Yields items found in the listing.

Parameters

- **limit** (*int*) – The maximum amount of items to search. If None, all are returned.
- **kwargs** (***Dict*) – Query parameters to append to the request URL.

Yields

- **subreddit** (*Subreddit*) – The subreddit found in the listing.
- **comment** (*Comment*) – The comment found in the listing.
- **submission** (*Submission*) – The submission found in the listing.
- **mod_action** (*ModAction*) – The mod action found in the listing.
- **wikipage_revision** (*WikipageRevision*) – The wikipage revision found in the listing.
- **item** (*aPRAWBase*) – A model of the item's data if kind couldn't be identified.

stream (*skip_existing: bool = False*, ***kwargs*) → AsyncIterator[apraw.models.helpers.apraw_base.aPRAWBase]
Stream items from an endpoint.

Streams use the asyncio.sleep() call to wait in between requests. If no items are found, the wait time is double until max_wait has been reached, at which point it's reset to 1.

Parameters

- **skip_existing** (*bool*) – Whether to skip items made before the call.
- **kwargs** (***Dict*) – Query parameters to append to the request URL.

Yields

- **subreddit** (*Subreddit*) – The subreddit found in the listing.
- **comment** (*Comment*) – The comment found in the listing.
- **submission** (*Submission*) – The submission found in the listing.
- **mod_action** (*ModAction*) – The mod action found in the listing.
- **wikipage_revision** (*WikipageRevision*) – The wikipage revision found in the listing.
- **item** (*aPRAWBase*) – A model of the item's data if kind couldn't be identified.

aPRAWBase

`aPRAWBase` is the base class used by most Reddit models to self-assign data retrieved from respective endpoints. It is used by classes such as *`Submission`* and *`Comment`*.

class apraw.models.**aPRAWBase**(*reddit: Reddit, data: Dict[str, Any], kind: str = ''*)
The base class for Reddit models.

The `aPRAWBase` class stores data retrieved by the endpoints and automatically assigns it as attributes. Specific information about the aforementioned attributes can be found in the respective implementations such as *`Comment`*.

reddit: Reddit The *`Reddit`* instance with which requests are made.

data: Dict The data obtained from the /about endpoint.

kind: str The item's kind / type.

ItemModeration

`ItemModeration` is a utility class to aid in moderation comments, submissions and modmail. Specific implementations such as *`CommentModeration`* exist as well, and the base class may be used by certain models.

class apraw.models.**ItemModeration**(*reddit: Reddit*, *item: apraw.models.helpers.apraw_base.aPRAWBase*)
A helper class to moderate comments, submissions and modmail.

reddit: Reddit The *`Reddit`* instance with which requests are made.

fullname: str The ID prepended with the kind of the item this helper belongs to.

approve()
Approve the Reddit item.

Returns **resp** – The API response JSON.

Return type Dict

distinguish(*how: NewType.<locals>.new_type = 'yes'*, *sticky: bool = False*)
Distinguish the Reddit item.

Parameters

- **how** (*`"yes" or "no" or "admin" or "special"`*) – The type of distinguishment to be added to the item.
- **sticky** (*`bool, optional`*) – Whether the item should be stickied.

Returns **resp** – The API response JSON.

Return type Dict

ignore_reports()
Ignore reports on the Reddit item.

Returns **resp** – The API response JSON.

Return type Dict

remove()
Remove the Reddit item.

Returns **resp** – The API response JSON.

Return type Dict

undistinguish()
Undistinguish the Reddit item.

Returns **resp** – The API response JSON.

Return type Dict

unignore_reports()
Unignore previously ignored reports on the Reddit item.

Returns **resp** – The API response JSON.

Return type Dict

CHAPTER 3

Index

- genindex

Index

A

approve() (*apraw.models.CommentModeration method*), 30
approve() (*apraw.models.ItemModeration method*), 34
approve() (*apraw.models.SubmissionModeration method*), 24
aPRAWBase (*class in apraw.models*), 34
AuthenticatedUser (*class in apraw.models*), 9
author() (*apraw.models.Comment method*), 28
author() (*apraw.models.ModmailMessage method*), 18
author() (*apraw.models.Submission method*), 22

C

clear_vote() (*apraw.models.Comment method*), 28
clear_vote() (*apraw.models.Submission method*), 22
Comment (*class in apraw.models*), 26
comment() (*apraw.Reddit method*), 10
CommentModeration (*class in apraw.models*), 30
comments() (*apraw.models.Submission method*), 22
conversations() (*apraw.models.SubredditModmail method*), 18

D

delete() (*apraw.models.Comment method*), 28
delete() (*apraw.models.Submission method*), 22
distinguish() (*apraw.models.CommentModeration method*), 30
distinguish() (*apraw.models.ItemModeration method*), 34
distinguish() (*apraw.models.SubmissionModeration method*), 24
downvote() (*apraw.models.Comment method*), 28
downvote() (*apraw.models.Submission method*), 23

F

full_data() (*apraw.models.Comment method*), 28
full_data() (*apraw.models.ModmailConversation method*), 19
full_data() (*apraw.models.Submission method*), 23

G

get() (*apraw.models.ListingGenerator method*), 33
get_auth_session() (*apraw.models.User method*), 9
get_client_session() (*apraw.models.User method*), 9
get_listing() (*apraw.Reddit method*), 10
get_request() (*apraw.Reddit method*), 10

H

hide() (*apraw.models.Comment method*), 28
hide() (*apraw.models.Submission method*), 23

I

ignore_reports() (*apraw.models.CommentModeration method*), 30
ignore_reports() (*apraw.models.ItemModeration method*), 34
ignore_reports() (*apraw.models.SubmissionModeration method*), 24
info() (*apraw.Reddit method*), 11
ItemModeration (*class in apraw.models*), 34

K

Karma (*class in apraw.models*), 9
karma() (*apraw.models.AuthenticatedUser method*), 9

L

ListingGenerator (*class in apraw.models*), 32
lock() (*apraw.models.CommentModeration method*), 30
lock() (*apraw.models.SubmissionModeration method*), 25

M

mark_nsfw() (*apraw.models.Submission method*), 23

mark_nsfw() (*apraw.models.SubmissionModeration method*), 25
me() (*apraw.models.User method*), 9
message() (*apraw.models.Redditor method*), 32
message() (*apraw.models.Subreddit method*), 15
message() (*apraw.Reddit method*), 11
messages() (*apraw.models.ModmailConversation method*), 19
mod() (*apraw.models.ModAction method*), 17
ModAction (*class in apraw.models*), 17
moderated_subreddits() (*apraw.models.Redditor method*), 32
moderators() (*apraw.models.Subreddit method*), 15
ModmailConversation (*class in apraw.models*), 18
ModmailMessage (*class in apraw.models*), 18
morechildren() (*apraw.models.Submission method*), 23

O

owner() (*apraw.models.ModmailConversation method*), 19

P

post_request() (*apraw.Reddit method*), 11

R

Reddit (*class in apraw*), 10
Redditor (*class in apraw.models*), 31
redditor() (*apraw.models.SubredditModerator method*), 16
redditor() (*apraw.Reddit method*), 11
refresh() (*apraw.models.Comment method*), 29
remove() (*apraw.models.CommentModeration method*), 30
remove() (*apraw.models.ItemModeration method*), 34
remove() (*apraw.models.SubmissionModeration method*), 25
replies() (*apraw.models.Comment method*), 29

S

save() (*apraw.models.Comment method*), 29
save() (*apraw.models.Submission method*), 23
show_comment() (*apraw.models.CommentModeration method*), 30
spoiler() (*apraw.models.Submission method*), 23
spoiler() (*apraw.models.SubmissionModeration method*), 25
sticky() (*apraw.models.SubmissionModeration method*), 25
stream() (*apraw.models.ListingGenerator method*), 33
Submission (*class in apraw.models*), 19
submission() (*apraw.models.Comment method*), 29
submission() (*apraw.Reddit method*), 11
SubmissionModeration (*class in apraw.models*), 24
Subreddit (*class in apraw.models*), 12
subreddit() (*apraw.models.Comment method*), 29
subreddit() (*apraw.models.Karma method*), 10
subreddit() (*apraw.models.Submission method*), 23
subreddit() (*apraw.Reddit method*), 12
SubredditModeration (*class in apraw.models*), 16
SubredditModerator (*class in apraw.models*), 16
SubredditModmail (*class in apraw.models*), 18

U

undistinguish() (*apraw.models.CommentModeration method*), 31
undistinguish() (*apraw.models.ItemModeration method*), 34
undistinguish() (*apraw.models.SubmissionModeration method*), 25
unhide() (*apraw.models.Comment method*), 29
unhide() (*apraw.models.Submission method*), 23
unignore_reports() (*apraw.models.CommentModeration method*), 31
unignore_reports() (*apraw.models.ItemModeration method*), 35
unignore_reports() (*apraw.models.SubmissionModeration method*), 25
unlock() (*apraw.models.CommentModeration method*), 31
unlock() (*apraw.models.SubmissionModeration method*), 25
unmark_nsfw() (*apraw.models.Submission method*), 24
unmark_nsfw() (*apraw.models.SubmissionModeration method*), 25
unsave() (*apraw.models.Comment method*), 29
unsave() (*apraw.models.Submission method*), 24
unspoiler() (*apraw.models.Submission method*), 24
unspoiler() (*apraw.models.SubmissionModeration method*), 26
unsticky() (*apraw.models.SubmissionModeration method*), 26
upvote() (*apraw.models.Comment method*), 30
upvote() (*apraw.models.Submission method*), 24
User (*class in apraw.models*), 8