chat-downloader Documentation

Release 0.2.0

xenova

May 20, 2022

CONTENTS

Chat Downloader is a simple tool used to retrieve chat messages from livestreams, videos, clips and past broadcasts. No authentication needed!

CHAPTER

ONE

INSTALLATION

This tool is distributed on PyPI and can be installed with `pip`:

```
$ pip install chat-downloader
```

To update to the latest version, add the `--upgrade` flag to the above command.

Alternatively, the tool can be installed with `git`:

```
$ git clone https://github.com/xenova/chat-downloader.git
$ cd chat-downloader
$ python setup.py install
```

CHAPTER

TWO

USAGE

2.1 Command line

```
usage: chat_downloader [-h] [--version] [--start_time START_TIME]
                       [--end_time END_TIME]
                       [--message_types MESSAGE_TYPES | --message_groups MESSAGE_GROUPS]
                       [--max_attempts MAX_ATTEMPTS]
                       [--retry_timeout RETRY_TIMEOUT]
                       [--interruptible_retry [INTERRUPTIBLE_RETRY]]
                       [--max_messages MAX_MESSAGES]
                       [--inactivity_timeout INACTIVITY_TIMEOUT]
                       [--timeout TIMEOUT] [--format FORMAT]
                       [--format_file FORMAT_FILE] [--chat_type {live,top}]
                       [--ignore IGNORE]
                       [--message_receive_timeout MESSAGE_RECEIVE_TIMEOUT]
                       [--buffer_size BUFFER_SIZE] [--output OUTPUT]
                       [--overwrite [OVERWRITE]] [--sort_keys [SORT_KEYS]]
                       [--indent INDENT] [--pause_on_debug | --exit_on_debug]
                       [--logging {none,debug,info,warning,error,critical} | --testing |␣
↪--verbose | --quiet]
                       [--cookies COOKIES] [--proxy PROXY]
                       url
```

For example, to save messages from a livestream to a JSON file, you can use:

```
$ chat_downloader https://www.youtube.com/watch?v=5qap5aO4i9A --output chat.json
```

For a description of these options, as well as advanced command line use-cases and examples, consult the *Command Line Usage* page.

2.2 Python

```
from chat_downloader import ChatDownloader

url = 'https://www.youtube.com/watch?v=5qap5aO4i9A'
chat = ChatDownloader().get_chat(url)       # create a generator
for message in chat:                        # iterate over messages
    chat.print_formatted(message)           # print the formatted message
```

For advanced python use-cases and examples, consult the *Python Documentation*.

CHAPTER

THREE

CHAT ITEMS

Chat items/messages are parsed into JSON objects (a.k.a. dictionaries) and should follow a format similar to this:

```
{
    ...
    "message_id": "xxxxxxxxxx",
    "message": "actual message goes here",
    "message_type": "text_message",
    "timestamp": 1613761152565924,
    "time_in_seconds": 1234.56,
    "time_text": "20:34",
    "author": {
        "id": "UCxxxxxxxxxxxxxxxxxxxxxx",
        "name": "username_of_sender",
        "images": [
            ...
        ],
        "badges": [
            ...
        ]
    },
    ...
}
```

For an extensive, documented list of included fields, consult the *Chat Item Fields* page.

CHAPTER

FOUR

FREQUENTLY ASKED QUESTIONS

Coming soon

CHAPTER

FIVE

ISSUES

Found a bug or have a suggestion? File an issue here. To assist the developers in fixing the issue, please follow the issue template as closely as possible.

CHAPTER

SIX

CONTRIBUTING

If you would like to help improve the tool, you'll find more information on contributing in our *Contributing Guide*.

CHAPTER

SEVEN

SUPPORTED SITES:

- YouTube.com - Livestreams, past broadcasts and premieres.
- Twitch.tv - Livestreams, past broadcasts and clips.
- Reddit.com - Livestreams, past broadcasts
- Facebook.com (currently in development) - Livestreams and past broadcasts.

CHAPTER

EIGHT

OVERVIEW

8.1 Command Line Usage

8.1.1 Overview

A full list of command line arguments can be obtained by running the help command:

```
$ chat_downloader -h
```

The output of which is as follows:

```
usage: chat_downloader [-h] [--version] [--start_time START_TIME]
                       [--end_time END_TIME]
                       [--message_types MESSAGE_TYPES | --message_groups MESSAGE_GROUPS]
                       [--max_attempts MAX_ATTEMPTS]
                       [--retry_timeout RETRY_TIMEOUT]
                       [--interruptible_retry [INTERRUPTIBLE_RETRY]]
                       [--max_messages MAX_MESSAGES]
                       [--inactivity_timeout INACTIVITY_TIMEOUT]
                       [--timeout TIMEOUT] [--format FORMAT]
                       [--format_file FORMAT_FILE] [--chat_type {live,top}]
                       [--ignore IGNORE]
                       [--message_receive_timeout MESSAGE_RECEIVE_TIMEOUT]
                       [--buffer_size BUFFER_SIZE] [--output OUTPUT]
                       [--overwrite [OVERWRITE]] [--sort_keys [SORT_KEYS]]
                       [--indent INDENT] [--pause_on_debug | --exit_on_debug]
                       [--logging {none,debug,info,warning,error,critical} | --testing |␣
↪--verbose | --quiet]
                       [--cookies COOKIES] [--proxy PROXY]
                       url

A simple tool used to retrieve chat messages from livestreams, videos, clips
and past broadcasts. No authentication needed!

Mandatory Arguments:
  url                   The URL of the livestream, video, clip or past
                        broadcast, defaults to None

General Arguments:
  -h, --help            show this help message and exit
```

(continues on next page)

(continued from previous page)

```
  --version             show program's version number and exit

Timing Arguments:
  --start_time START_TIME, -s START_TIME
                        Start time in seconds or hh:mm:ss, defaults to None
                        (as early as possible)
  --end_time END_TIME, -e END_TIME
                        End time in seconds or hh:mm:ss, defaults to None
                        (until the end)

Message Type Arguments:
  --message_types MESSAGE_TYPES
                        List of messages types to include, defaults to None
  --message_groups MESSAGE_GROUPS
                        List of messages groups (a predefined, site-specific
                        collection of message types) to include

Retry Arguments:
  --max_attempts MAX_ATTEMPTS
                        Maximum number of attempts to retrieve chat messages,
                        defaults to 15
  --retry_timeout RETRY_TIMEOUT
                        Number of seconds to wait before retrying. Setting
                        this to a negative number will wait for user input.
                        Default is None (use exponential backoff, i.e.
                        immediate, 1s, 2s, 4s, 8s, ...)
  --interruptible_retry [INTERRUPTIBLE_RETRY]
                        Have the option to skip waiting and immediately retry.
                        Defaults to True

Termination Arguments:
  --max_messages MAX_MESSAGES
                        Maximum number of messages to retrieve, defaults to
                        None (unlimited)
  --inactivity_timeout INACTIVITY_TIMEOUT
                        Stop getting messages after not receiving anything for
                        a certain duration (in seconds), defaults to None
  --timeout TIMEOUT     Stop retrieving chat after a certain duration (in
                        seconds), defaults to None

Format Arguments:
  --format FORMAT       Specify how messages should be formatted for printing,
                        defaults to the site's default value
  --format_file FORMAT_FILE
                        Specify the path of the format file to choose formats
                        from, defaults to None

[Site Specific] YouTube Arguments:
  --chat_type {live,top}
                        Specify chat type, defaults to 'live'
  --ignore IGNORE       Ignore a list of video ids, defaults to None
```

(continues on next page)

(continued from previous page)

```
[Site Specific] Twitch Arguments:
  --message_receive_timeout MESSAGE_RECEIVE_TIMEOUT
                        Time before requesting for new messages, defaults to
                        0.1
  --buffer_size BUFFER_SIZE
                        Specify a buffer size for retrieving messages,
                        defaults to 4096

Output Arguments:
  --output OUTPUT, -o OUTPUT
                        Path of the output file, defaults to None (print to
                        standard output)
  --overwrite [OVERWRITE]
                        If True, overwrite output file. Otherwise, append to
                        the end of the file. Defaults to True. In both cases,
                        the file (and directories) is created if it does not
                        exist.
  --sort_keys [SORT_KEYS]
                        Sort keys when outputting to a file, defaults to True
  --indent INDENT       Number of spaces to indent JSON objects by. If
                        nonnumerical input is provided, this will be used to
                        indent the objects. Defaults to 4

Debugging/Testing Arguments:
  --pause_on_debug      Pause on certain debug messages, defaults to False
  --exit_on_debug       Exit when something unexpected happens, defaults to
                        False
  --logging {none,debug,info,warning,error,critical}
                        Level of logging to display, defaults to info
  --testing             Enable testing mode. This is equivalent to setting
                        logging to debug and enabling pause_on_debug. Defaults
                        to False
  --verbose, -v         Print various debugging information. This is
                        equivalent to setting logging to debug. Defaults to
                        False
  --quiet, -q           Activate quiet mode (hide all output), defaults to
                        False

Initialisation Arguments:
  --cookies COOKIES, -c COOKIES
                        Path of cookies file, defaults to None
  --proxy PROXY, -p PROXY
                        Use the specified HTTP/HTTPS/SOCKS proxy. To enable
                        SOCKS proxy, specify a proper scheme. For example
                        socks5://127.0.0.1:1080/. Pass in an empty string
                        (--proxy "") for direct connection. Defaults to None
```

8.1.2 Examples

1. Message groups and types

   Options are specified by a space- or comma-separated list. If you specify more than one item, enclose the argument in quotation marks.

   - Using message groups:

```
$ chat_downloader https://www.youtube.com/watch?v=n5aQeLwwEns --message_groups
↪"messages superchat"
```

   - Using message types:

```
$ chat_downloader https://www.youtube.com/watch?v=n5aQeLwwEns --message_types␣
↪membership_item
```

2. Output to file

```
$ chat_downloader https://www.youtube.com/watch?v=n5aQeLwwEns --output chat.json
```

8.2 Python Documentation

8.2.1 ChatDownloader

class chat_downloader.**ChatDownloader**(*headers=None*, *cookies=None*, *proxy=None*)

Bases: `object`

Class used to create sessions and download chats.

__init__(*headers=None*, *cookies=None*, *proxy=None*)

Initialise a new session for making requests. Parameters are saved and are sent to the relevant constructor when creating a new session.

Parameters

- **headers** (*dict, optional*) – Headers to use for subsequent requests, defaults to None
- **cookies** (*str, optional*) – Path of cookies file, defaults to None
- **proxy** (*str, optional*) – Use the specified HTTP/HTTPS/SOCKS proxy. To enable SOCKS proxy, specify a proper scheme. For example socks5://127.0.0.1:1080/. Pass in an empty string (–proxy “”) for direct connection. Defaults to None

close()

Close all sessions associated with the object

create_session(*chat_downloader_class*, *overwrite=False*)

get_chat(*url=None*, *start_time=None*, *end_time=None*, *max_attempts=15*, *retry_timeout=None*, *interruptible_retry=True*, *timeout=None*, *inactivity_timeout=None*, *max_messages=None*, *message_groups=<chat_downloader.sites.common.SiteDefault object>*, *message_types=None*, *output=None*, *overwrite=True*, *sort_keys=True*, *indent=4*, *format=<chat_downloader.sites.common.SiteDefault object>*, *format_file=None*, *chat_type='live'*, *ignore=None*, *message_receive_timeout=0.1*, *buffer_size=4096*)

Used to get chat messages from a livestream, video, clip or past broadcast.

Parameters

- **url** (*str, optional*) – The URL of the livestream, video, clip or past broadcast, defaults to None
- **start_time** (*float, optional*) – Start time in seconds or hh:mm:ss, defaults to None (as early as possible)
- **end_time** (*float, optional*) – End time in seconds or hh:mm:ss, defaults to None (until the end)
- **max_attempts** (*int, optional*) – Maximum number of attempts to retrieve chat messages, defaults to 15
- **retry_timeout** (*float, optional*) – Number of seconds to wait before retrying. Setting this to a negative number will wait for user input. Default is None (use exponential backoff, i.e. immediate, 1s, 2s, 4s, 8s, ...)
- **interruptible_retry** (*bool, optional*) – Have the option to skip waiting and immediately retry. Defaults to True
- **timeout** (*float, optional*) – Stop retrieving chat after a certain duration (in seconds), defaults to None
- **inactivity_timeout** (*float, optional*) – Stop getting messages after not receiving anything for a certain duration (in seconds), defaults to None
- **max_messages** (*int, optional*) – Maximum number of messages to retrieve, defaults to None (unlimited)
- **message_groups** (*SiteDefault, optional*) – List of messages groups (a predefined, site-specific collection of message types) to include
- **message_types** (*list, optional*) – List of messages types to include, defaults to None
- **output** (*str, optional*) – Path of the output file, defaults to None (print to standard output)
- **overwrite** (*bool, optional*) – If True, overwrite output file. Otherwise, append to the end of the file. Defaults to True. In both cases, the file (and directories) is created if it does not exist.
- **sort_keys** (*bool, optional*) – Sort keys when outputting to a file, defaults to True
- **indent** (*Union[int, str], optional*) – Number of spaces to indent JSON objects by. If nonnumerical input is provided, this will be used to indent the objects. Defaults to 4
- **format** (*SiteDefault, optional*) – Specify how messages should be formatted for printing, defaults to the site's default value
- **format_file** (*str, optional*) – Specify the path of the format file to choose formats from, defaults to None
- **chat_type** (*str, optional*) – Specify chat type, defaults to 'live'
- **ignore** (*list, optional*) – Ignore a list of video ids, defaults to None
- **message_receive_timeout** (*float, optional*) – Time before requesting for new messages, defaults to 0.1
- **buffer_size** (*int, optional*) – Specify a buffer size for retrieving messages, defaults to 4096

Raises

- **URLNotProvided** – if no URL is provided
- **ChatGeneratorError** – if no valid generator can be found for a site
- **SiteNotSupported** – if no matching site can be found
- **InvalidURL** – if the URL provided is not valid

Returns The appropriate Chat object, given these parameters

Return type Chat

get_session(*chat_downloader_class*)

Examples

1. Message groups and types

   Options are specified as a list of strings.

```
from chat_downloader import ChatDownloader

downloader = ChatDownloader()

url = 'https://www.youtube.com/watch?v=n5aQeLwwEns'

# 1. Using message groups:
groups_example = downloader.get_chat(url, message_groups=['messages', 'superchat'])

# 2. Using message types:
types_example = downloader.get_chat(url, message_types=['membership_item'])
```

2. 2

8.2.2 sites Module

YouTubeChatDownloader

class chat_downloader.sites.**YouTubeChatDownloader**(***kwargs*)

Bases: *chat_downloader.sites.common.BaseChatDownloader*

__init__(***kwargs*)

Initialise a session with various parameters

Raises **CookieError** – if unable to read or parse the cookie file

generate_urls(***kwargs*)

This method should be implemented in a subclass and should return a generator which yields URLs for testing.

Raises **NotImplementedError** – if not implemented and called from a subclass

get_chat_by_channel_id(*channel_id*, *params*)

get_chat_by_clip_id(*clip_id*, *params*)

get_chat_by_custom_username(*custom_username*, *params*)

`get_chat_by_user_id`(*user_id*, *params*)

Such as NASAtelevision in https://www.youtube.com/user/NASAtelevision

Parameters **user_id** (*[type]*) –

`get_chat_by_video_id`(*video_id*, *params*)

Get chat messages for a YouTube video, given its ID.

Parameters **video_id** (*str*) – YouTube video ID

Returns Chat object for the corresponding YouTube video

Return type Chat

`get_playlist_items`(*playlist_url*, *params=None*)

`get_user_videos`(*channel_id=None*, *user_id=None*, *custom_username=None*, *video_status='all'*, *params=None*)

Retrieve all videos listed on the user's channel

If more than one of *channel_id*, *user_id* and *custom_username* are specifed, the first one specified will be returned.

Parameters

- **channel_id** (*str, optional*) – The user's channel ID, defaults to None. (e.g., https://www.youtube.com/channel/<channel_id>)
- **user_id** (*str, optional*) – The user's ID, defaults to None (e.g., https://www.youtube.com/user/<user_id>)
- **custom_username** (*str, optional*) – [description], defaults to None (e.g., https://www.youtube.com/c/<custom_username>)
- **video_status** (*str, optional*) – Determines which videos will be retrieved, defaults to 'all'. Must be one of 'all', 'live', 'upcoming' or 'past'.
- **params** (*dict, optional*) – Additional program parameters, defaults to None

Raises

- **ValueError** – If no user is specified or an invalid video_status is specified
- **UserNotFound** – If the user cannot be found
- **NoVideos** – If the channel has no videos

Yield The next video

Return type dict

`get_video_data`(*video_id*, *params=None*)

TwitchChatDownloader

class chat_downloader.sites.**TwitchChatDownloader**(***kwargs*)

Bases: *chat_downloader.sites.common.BaseChatDownloader*

__init__(***kwargs*)

Initialise a session with various parameters

Raises **CookieError** – if unable to read or parse the cookie file

generate_urls(*livestream_limit*, *vod_limit*, *clip_limit*, ***kwargs*)

This method should be implemented in a subclass and should return a generator which yields URLs for testing.

Raises **NotImplementedError** – if not implemented and called from a subclass

get_chat_by_clip_id(*clip_id*, *params*)

get_chat_by_stream_id(*stream_id*, *params*)

get_chat_by_vod_id(*vod_id*, *params*)

get_featured_videos(*username*)

get_top_livestreams(*limit=30*)

get_user_clips(*username*, *limit=100*, *filter_by='LAST_WEEK'*)

get_user_videos(*username*, *limit=None*, *video_type=None*, *sort='TIME'*)

FacebookChatDownloader

BaseChatDownloader

class chat_downloader.sites.**BaseChatDownloader**(***kwargs*)

Bases: `object`

Base class for chat downloaders. Each supported site should have its own chat downloader. Subclasses should redefine the *_VALID_URLS* dictionary which creates a mapping between functions and their matching regular expressions. Optionally, subclasses should also redefine *_NAME*, *_SITE_DEFAULT_PARAMS* and *_TESTS* fields.

__init__(***kwargs*)

Initialise a session with various parameters

Raises **CookieError** – if unable to read or parse the cookie file

static check_for_invalid_types(*messages_types_to_add*, *allowed_message_types*)

Used to check for invalid message types

Parameters

- **messages_types_to_add** (*list*) – List of message types to add
- **allowed_message_types** (*list*) – List of allowed message type

Raises **InvalidParameter** – if invalid types are specified

clear_cookies()

Clear the session's cookies.

close()

Close the session. Once this has been called, no more requests can be made.

generate_urls(***kwargs*)

This method should be implemented in a subclass and should return a generator which yields URLs for testing.

Raises NotImplementedError – if not implemented and called from a subclass

get_cookie_value(*name*, *default=None*)

Return the value for key if key is in the cookie dictionary, else default.

Parameters

- **name** (*str*) – The key of the cookie
- **default** (*object, optional*) – Return this value if the specified cookie cannot be found, defaults to None

Returns The cookie value, or default

Return type Union[str, object, None]

static get_mapped_keys(*remapping*)

get_session_headers(*key*)

get_site_value(*value*)

Get the site's default value for a certain parameter

Parameters **value** (*Union[SiteDefault, object]*) – The value

Returns The site's default value

Return type object

classmethod matches(*url*)

Used to check if a url matches any of the regular expressions specified in the classes *_VALID_URLS* dictionary.

Returns If a match is found, the function name and match object is returned, otherwise None.

Return type (str, re.Match)

static retry(*attempt_number*, *max_attempts=1*, *error=None*, *retry_timeout=None*, *text=None*, *interruptible_retry=True*, ***kwargs*)

Retry to occur after an error occurs

Parameters

- **attempt_number** (*int*) – The current attempt number
- **max_attempts** (*int, optional*) – The maximum number of attempts allowed
- **error** (*Exception, optional*) – The error which was raised, defaults to None
- **retry_timeout** (*float, optional*) – The number of seconds to sleep after failing, defaults to None (i.e. use exponential back-off)
- **text** (*object, optional*) – Items to display on retry, defaults to None

Raises RetriesExceeded – if the maximum number of retries has been exceeded

set_cookie_value(*domain*, *name*, *value*, *expire_time=None*, *port=None*, *path='/'*, *secure=False*, *discard=False*, *rest={}*, ***kwargs*)

update_session_headers(*new_headers*)

8.3 Chat Item Fields

First of all, each chat item is a dictionary. The following tables list and provide explanations for the various fields that a chat item may possess.

If you find a field that is not listed below, please notify the developers by creating an issue, or adding the relevant documentation in a pull request.

Note: It is recommended to treat every field listed below as optional. While most items contain basic information such as *timestamp*, *message* or *author*, it cannot be guaranteed that every item will contain these fields.

Table 1: Common fields

Field	Type	Description
timestamp	float	UNIX time (in microseconds) of when the message was sent.
message	string	Actual content/text of the chat item.
message_id	string	Identifier for the chat item.
message_type	string	Message type of the item.
author	dictionary	A dictionary containing information about the user who sent the message. For author fields, see *here*.
time_in_seconds	float	The number of seconds after the video began, that the message was sent. This is only present for replays/vods/clips (i.e. a video which is not live).
time_text	string	Human-readable format for *time_in_seconds*.

Documentation for other (less common) fields can be found *here*.

Table 2: Author fields

Field	Type	Description
name	string	The name of the author.
id	string	Idenfifier for the author.
display_name	string	The name of the author which is displayed to the viewer. This may be different to *name*.
short_name	string	A shortened version of the author's name.
type	string	Type of the author.
url	string	URL of the author's channel/page.
images	list	A list which contains different sizes of the author's profile picture. See *here* for the fields that an image may have.
badges	list	A list of the author's badges. See *here* for the fields that a badge may have.
gender	string	Gender of the author.
is_banned	boolean	*True* if the user is banned, *False* otherwise.
is_bot	boolean	*True* if the user is a bot, *False* otherwise.
is_non_coworker	boolean	*True* if the user is not a coworker, *False* otherwise.
is_original_poster	boolean	*True* if the user is the original poster, *False* otherwise.
is_verified	boolean	*True* if the user is verified, *False* otherwise.

Table 3: Image fields

Field	Type	Description
url	string	The URL of the actual image
width	integer	The width of the image
height	integer	The height of the image
image_id	string	A identifier for the image, usually of the form: {width}x{height}

Table 4: Badge fields

Field	Type	Description
title	string	The title of the badge.
id	string	Identifier for the badge.
name	string	Name of the badge.
version	integer	Version of the badge.
icon_name	string	Name of the badge icon.
icons	list	A list of images for the badge icons. See *here* for potential fields.
description	string	The description of the badge.
alternative_title	string	Alternative title of the badge.
click_action	string	Action to perform if the badge is clicked.
click_url	string	URL to visit if the badge is clicked.

Table 5: Other fields

Field	Type	Description
amount	float	The amount of money that was sent with the message.
sub_message	string	Additional text of the message.
action_type	string	Action type of the item.
tooltip	string	Text to be displayed when hovering over the message.
icon	string	Icon associated with the message.
target_message_id	string	The identifier for a message which this message references.
action	string	The action of the message.
viewer_is_creator	boolean	Whether the viewer is the creator or not.
sticker_images	list	A list which contains different sizes of the sticker image. See *here* for image fields.
sponsor_icons	list	A list which contains different sizes of the sponsor image. See *here* for image fields.
ticker_icons	list	A list which contains different sizes of the ticker image. See *here* for image fields.
ticker_duration	float	How long the ticker message is displayed for.
field	type	description
field	type	description
field	type	description
field	type	description
field	type	description
field	type	description

The following fields indicate HEX colour information for the message:

author_name_text_colour timestamp_colour body_background_colour header_text_colour header_background_colour body_text_colour background_colour money_chip_text_colour money_chip_background_colour start_background_colour amount_text_colour end_background_colour detail_text_colour

8.4 General Options

This page provides descriptions for the different available options. For the most part, there are two types of options:

1. Initialization - These options are used when creating a new `ChatDownloader` object. All subsequent `get_chat` method calls using this object will use these options.
2. Program - These options are passed to the object's `get_chat` method. These options are not shared between other calls to `get_chat`.

For implementation details and examples, see the corresponding Command-line or Python usage guide.

8.4.1 Initialization Options

8.4.2 Program Options

Message groups and types

There are two ways to specify what types of chat messages will be included:

1. Message types - The value of *message_type* for the chat object
2. Message groups - A site-specific collection of message types. One message group contains a list of message types, and each site has a number of message groups which can be used. See below for message groups (and their associated message types) for the supported sites.

Please note that these two options are **mutually exclusive**. So, you may only specify one at a time.

Message groups and types for the supported sites are specified as follows:

```
site.com
--------
 - message_group_1
   - message_type_1
   - message_type_2
 - message_group_2
   - message_type_3
   - message_type_4
```

For example, specifying *message_group_1* as a message group will include all messages whose type is *message_type_1* or *message_type_2*. Alternatively, one may specify individual message types, e.g. *message_type_3*.

The following message groups and message types are allowed for each supported site:

```
youtube.com
-----------
- messages
  - text_message
- superchat
  - membership_item
  - paid_message
  - paid_sticker
- tickers
  - ticker_paid_sticker_item
  - ticker_paid_message_item
  - ticker_sponsor_item
```

(continues on next page)

(continued from previous page)

```
- banners
  - banner
  - banner_header
- donations
  - donation_announcement
- engagement
  - viewer_engagement_message
- purchases
  - purchased_product_message
- mode_changes
  - mode_change_message
- deleted
  - deleted_message
- bans
  - ban_user
- placeholder
  - placeholder_item

twitch.tv
---------
- messages
  - text_message
  - highlighted_message
  - send_message_in_subscriber_only_mode
- bans
  - ban_user
  - already_banned
  - bad_ban_self
  - bad_ban_broadcaster
  - bad_ban_admin
  - bad_ban_global_mod
  - bad_ban_staff
  - ban_success
  - bad_unban_no_ban
  - unban_success
  - channel_suspended_message
  - timeout_success
  - bad_timeout_self
  - bad_timeout_broadcaster
  - bad_timeout_mod
  - bad_timeout_admin
  - bad_timeout_global_mod
  - bad_timeout_staff
- deleted_messages
  - delete_message
  - banned_message
  - bad_delete_message_error
  - bad_delete_message_broadcaster
  - bad_delete_message_mod
  - delete_message_success
- hosts
  - host_target
```

(continues on next page)

(continued from previous page)

```
  - start_host
  - end_host
  - bad_host_hosting
  - bad_host_rate_exceeded
  - bad_host_error
  - hosts_remaining
  - not_hosting
  - host_target_went_offline
- room_states
  - room_state
  - enable_slow_mode
  - disable_slow_mode
  - slow_mode_already_on
  - slow_mode_already_off
  - enable_subscriber_only_mode
  - disable_subscriber_only_mode
  - sub_mode_already_on
  - sub_mode_already_off
  - enable_emote_only_mode
  - disable_emote_only_mode
  - emote_only_already_on
  - emote_only_already_off
  - enable_r9k_mode
  - disable_r9k_mode
  - r9k_mode_already_on
  - r9k_mode_already_off
  - enable_follower_only_mode
  - enable_follower_only_mode
  - disable_follower_only_mode
  - follower_only_mode_already_on
  - follower_only_mode_already_on
  - follower_only_mode_already_off
- user_states
  - user_state
- notices
  - user_notice
  - notice
  - successful_login
- chants
  - crowd_chant
  - crowd_chant
- other
  - clear_chat
  - reconnect
  - cmds_available
  - unrecognized_cmd
  - no_permission
  - rate_limit_reached_message
- bits
  - bits_badge_tier
- subscriptions
  - subscription
```

(continues on next page)

(continued from previous page)

```
  - resubscription
  - subscription_gift
  - anonymous_subscription_gift
  - anonymous_mystery_subscription_gift
  - mystery_subscription_gift
  - extend_subscription
  - standard_pay_forward
  - community_pay_forward
  - prime_community_gift_received
- upgrades
  - prime_paid_upgrade
  - gift_paid_upgrade
  - reward_gift
  - anonymous_gift_paid_upgrade
- raids
  - raid
  - unraid
- rituals
  - ritual
- mods
  - bad_mod_banned
  - bad_mod_mod
  - mod_success
  - bad_unmod_mod
  - unmod_success
  - no_mods
  - room_mods
- colours
  - turbo_only_colour
  - colour_changed
- commercials
  - bad_commercial_error
  - commercial_success
- vips
  - bad_vip_grantee_banned
  - bad_vip_grantee_already_vip
  - vip_success
  - bad_unvip_grantee_not_vip
  - unvip_success
  - no_vips
  - vips_success
- charity
  - charity

reddit.com
----------
```

Outputting to a file

8.5 Contributing Guide

8.5.1 Developers

The following section outlines the basic procedure for people who want to assist in development.

Add features, fix bugs or write documentation

1. Fork this repository.
2. Clone the forked repository with:

```
$ git clone git@github.com:YOUR_GITHUB_USERNAME/chat-downloader.git
```

3. Start a new branch with:

```
$ cd chat-downloader
$ git checkout -b name
```

4. Set up your environment by installing the developer dependencies:

```
$ pip install -e .[dev]
```

5. Make changes. See below for common changes to be made.
6. Test your changes with pytest:

   Depending on the changes made, run the appropriate tests to ensure everything still works as intended.

   1. To run all tests:

   ```
   $ pytest -v
   ```

   2. To run tests for all sites:

   ```
   $ pytest -v tests/test_chat_downloader.py::TestChatDownloader
   ```

   3. To run the tests for a specific site:

   ```
   $ pytest -v tests/test_chat_downloader.py::TestChatDownloader::test_
   ↪YourChatDownloader_TestNumber
   ```

   e.g.

   ```
   $ pytest -v tests/test_chat_downloader.py::TestChatDownloader::test_
   ↪YouTubeChatDownloader_1
   ```

7. Make sure your code follows our coding conventions and check the code with flake8:

```
$ flake8 path/to/code/to/check.py
```

While we encourage users to follow flake8 conventions, some warnings are not very important and can be ignored, e.g:

```
$ flake8 path/to/code/to/check.py --ignore E501,W503,W504
```

8. When the tests pass, add the new files and commit them and push the result, like this:

```
$ git add path/to/code.py
$ git commit -m 'message'
$ git push origin name
```

9. Finally, create a pull request. We'll then review and merge it.

Starting templates

When adding new features, we encourage developers to use these templates as starting points. This helps ensure consistency across the codebase.

New site

Coming soon

8.5.2 Testing

If you are unable to write code but still wish to assist, we encourage users to run commands with the `--testing` flag included. This will print debugging messages and pause once something unexpected happens (e.g. when an unknown item is being parsed). If something happens, please raise an issue and we will fix it or add support for it as soon as possible! For example:

```
$ chat_downloader https://www.youtube.com/watch?v=5qap5aO4i9A --testing
```

Some extractors use undocumented endpoints and as a result, users may encounter items which will not be parsed correctly. Increased testing will help find these items and ultimately improve functionality of the software for other users. Note that this will not affect any output you write to files (using `--output`).

8.5.3 Sponsor

Chat Downloader has always and will always be free. If you are feeling generous, donations are always appreciated!

- https://ko-fi.com/xenova
- https://www.buymeacoffee.com/xenova

8.6 Changelog

8.6.1 master

8.6.2 v0.0.4

Date 11 February 2021

Sites

- Use `_NAME` attribute and improve class structure
- [YouTube] Unpack value returned from `parse_runs` (Fixes #59)
- [Twitch] Move `is_moderator`, `is_subscriber` and `is_turbo` to author dictionary

8.6.3 v0.0.3

Date 10 February 2021

Core

- Allow reusing of sessions

Sites

- Improved remapping (more advanced)
- YouTube and Twitch: Parse emotes to get emote id and URLs

Testing

- Improved unit testing with pytest

Workflows

- CI with GitHub actions
- Automatic distribution with twine on release

8.6.4 v0.0.2

Date 3 February 2021

- Ensure mutual exclusion for message groups and types

8.6.5 v0.0.1

Date 2 February 2021

- Initial release

PYTHON MODULE INDEX

C

INDEX

Symbols

B

C

G

M

R

S

T

U

Y