iFunny

Release 0.11.1

Sep 21, 2020

Contents:

Requirements:

- requests
- websocket-client
- Python 3.7+

This is a python library aiming to interface with python. To learn more about the project, and to view it's code, check it out on github.

CHAPTER 1

About

This library can do a number of things in the scope of interacting with iFunny, though it is still very much in development and not (even close to) everything is implemented. This library is also able to interface with ifunny chat (which just looks to be a modified client of sendbird) in a way idomatic to Python.

Though you can interface with raw responses, this lib provides a number of decorators for chat events (with more on the way), as well as decorators for commands executed with prefixes (inspired by discordpy). The chat client is ran in it's own thread (unless specified not to, see debugging docs) and each messages triggers an event (again, in a separate thread)

CHAPTER 2

Examples

To create a simple chat bot, the most important steps are as follows - create a client and authenticate it - create a command method and decorate it - start the chat thread (you can do this in any order, but it's probably best if you create your commands first)

A simple echo bot might look like this:

```
from ifunny import Client
robot = Client(prefix = "/")
robot.login("email", "password")

@robot.event(name = "on_connect")
def _connected_to_chat(data):
    print("I'm connected")

@robot.command(name = "echo")
def _reply_with_same(message, args):
    message.send(f"You said {message.content}")

robot.start_chat()
```

There's still much more to do, so feel free to create a pull. If you want to find me, my discord is Zero#5200. You can also join a guild with me here.

2.1 ClientBase

class ifunny.objects._mixin.**ClientBase**(*paginated_size=25*, *captcha_api_key=None*)

iFunny Client base class. Also used standalone for some read-only actions that do not warrant a Client that may log in

Parameters **paginated_size** (*int*) – default number of elemets to request for each paginated data call

Captcha_api_key 2captcha api key to use for attempts at creating accounts

captcha_api = 'https://2captcha.com'

new_basic_token
Generate a new basic token, even if one is stored

Returns Basic oauth2 token

Return type string

basic_token
Generate or load from config a Basic auth token.

Returns Basic oauth2 token

Return type str

headers
Generate headers for iFunny requests dependant on authentication

Returns request-ready headers

Return type dict

search_users(*query*)
Search for users

Parameters **query** (*str*) – query to search

Returns generator iterating search results

Return type generator<User>

search_tags(*query*)
Search for tags

Parameters **query** (*str*) – query to search

Returns generator iterating search results

Return type generator<Post>

search_chats(*query*)
Search for chats

Parameters **query** (*str*) – query to search

Returns generator iterating search results

Return type generator<Chat>

mark_features_read()
Mark featured feed as read (or viewed).

email_is_available(*email*)
Check email availability :param email: email in question

Returns is this email available?

Return type bool

nick_is_available(*nick*)
Check nick availability :param nick: nick in question

Returns is this nick available?

Return type bool

notifications
generator for a client's notifications. Each iteration will return the next notification, in decending order of date recieved

Returns generator iterating through notifications

Return type generator<Notification>

reads
generator for a client's reads. Each iteration will return the next viewed post, in decending order of date accessed

Returns generator iterating through read posts

Return type generator<Post>

viewed
Alias to Client.reads because ifunny's in-api name is dumb. You don't read a picture or video

collective
generator for the collective feed. Each iteration will return the next collective post, in decending order of date posted

Returns generator iterating the collective feed

Return type generator<Post>

featured
generator for the featured feed. Each iteration will return the next featured post, in decending order of date posted

Returns generator iterating the featured feed

Return type generator<Post>

digests

Returns digests available to the client from explore

Return type generator<Digest>

channels

Returns a list of channels featured in explore

Return type list<Channel>

trending_chats

Returns a list of trending chats featured in explore

Return type list<Chat>

messenger_token

counters

Returns ifunny unread counters

Return type dict

unread_featured

Returns unread featured posts

Return type int

unread_collective

Returns unread collective posts

Return type int

unread_subscriptions

Returns unread subscriptions posts

Return type int

unread_news

Returns unread news posts

Return type int

2.2 ObjectMixin

class ifunny.objects._mixin.**ObjectMixin**(*id*, *client=<ifunny.objects._mixin.ClientBase object>*, *data=None*, *paginated_size=30*)

Mixin class for iFunny objects. Used to implement common methods

Parameters

- **id** (*str*) – id of the object
- **client** (Client) – Client that the object belongs to
- **data** (*dict*) – A data payload for the object to pull from before requests
- **paginated_size** (*int*) – number of items to get for each paginated request. If above the call type's maximum, that will be used instead

get(*key*, *default=None*)

fresh

Returns self after setting the update flag

Return type Subclass of ObjectMixin

is_deleted

Returns is this object deleted?

Return type bool

headers

2.3 SendbirdMixin

class ifunny.objects._mixin.**SendbirdMixin**(*id*, *client=<ifunny.objects._mixin.ClientBase object>*, *data=None*, *paginated_size=30*)

Mixin class for sendbird objects. Used to implement common methods, subclass to ObjectMixin

Parameters

- **id** (*str*) – id of the object
- **client** (Client) – Client that the object belongs to
- **data** (*dict*) – A data payload for the object to pull from before requests

- **paginated_size** (*int*) – number of items to get for each paginated request. If above the call type's maximum, that will be used instead

2.4 Client

class ifunny.**Client**(*trace=False*, *threaded=True*, *prefix={"}*, *paginated_size=25*, *captcha_api_key=None*)

iFunny client used to do most things.

Parameters

- **trace** (*bool*) – enable websocket_client trace? (debug)
- **threaded** (*bool*) – False to have all socket callbacks run in the same thread for debugging
- **prefix** (*str or callable*) – Static string or callable prefix for chat commands
- **paginated_size** (*int*) – Number of items to request in paginated methods

get(*key*, *default=None*)

login(*email*, *password="*, *force=False*)

Authenticate with iFunny to get an API token. Will try to load saved account tokens (saved as plaintext json, indexed by *email_token*) if *force* is False

Parameters

- **email** (*str*) – Email associated with the account
- **password** (*str*) – Password associated with the account
- **force** (*bool*) – Ignore saved Bearer tokens?

Returns self

Return type *Client*

post_image_url(*image_url*, ***kwargs*)

Post an image from a url to iFunny

Parameters

- **image_url** – location image to post
- **tags** (*list<str>*) – list of searchable tags
- **visibility** (*str*) – Visibility of the post on iFunny. Can be one of (public, subscribers)
- **wait** (*bool*) – wait for the post to be successfuly published?
- **timeout** (*int*) – time to wait for a successful post
- **schedule** (*int, or None*) – timestamp to schedule the post for, or None for immediate

Returns Post if wait flag set (when posted)

Return type *Post*, or None

post_image(*image_data*, *tags=[]*, *visibility='public'*, *type='pic'*, *wait=False*, *timeout=15*, *schedule=None*)

Post an image to iFunny

Parameters

- **image_data** (*bytes*) – Binary image to post
- **tags** (*list<str>*) – List of searchable tags
- **visibility** (*str*) – Visibility of the post on iFunny. Can be one of (`public`, `subscribers`)
- **type** (*str*) – type of content to post. Can be one of (`pic`, `gif`)
- **wait** (*bool*) – wait for the post to be successfuly published?
- **timeout** (*int*) – time to wait for a successful post
- **schedule** (*int, or None*) – timestamp to schedule the post for, or None for immediate

Returns Post if wait flag set (when posted)

Return type *Post*, or None

resolve_command(*message*)
Find and call a command called from a message

Parameters **message** (`Message`) – Message object recieved from the sendbird socket

suggested_tags(*query*)
Tags suggested by ifunny for a query

Parameters **query** (*str*) – query for suggested tags

Returns list of suggested tags and the number of memes with it

Rty list<tuple<str, int>>

start_chat()
Start the chat websocket connection.

Returns this client's socket object

Return type Socket

Raises Exception stating that the socket is already alive

stop_chat()
Stop the chat websocket connection.

Returns this client's socket object

Return type Socket

sendbird_upload(*chat*, *file_data*)
Upload an image to sendbird for a specific chat

Parameters

- **chat** (`ifunny.objects.Chat`) – chat to upload the file for
- **file_data** (*bytes*) – binary file to upload

Returns url to the uploaded content

Return type str

command(*name=None*)
Decorator to add a command, callable in chat with the format `{prefix}{command}` Commands must take two arguments, which are set as the Message and list<str> of space-separated words in the message (excluding the command) respectively:

```
import ifunny
robot = ifunny.Client()

@robot.command()
def some_command(ctx, args):
    # do something
    pass
```

Parameters **name** (*str*) – Name of the command callable from chat. If None, the name of the function will be used instead.

event (*name=None*)
Decorator to add an event, which is called when different things happen by the clients socket. Events must take one argument, which is a dict with the websocket data:

```
import ifunny
robot = ifunny.Client()

@robot.event(name = "on_connect")
def event_when_connected_to_chat(data):
    print(f"{robot} is chatting")
```

Parameters **name** (*str*) – Name of the event. If None, the name of the function will be used instead. See the Sendbird section of the docs for valid events.

sendbird_headers
Generate headers for a sendbird api call. If a messenger_token exists, it's added

Returns sendbird-ready headers

Return type dict

headers
Generate headers for iFunny requests dependant on authentication

Returns request-ready headers

Return type dict

prefix
Get a set of prefixes that this bot can use. Each one is evaluated when handling a potential command

Returns prefixes that can be used to resolve commands

Return type set

messenger_token
Get the messenger_token used for sendbird api calls If a value is not stored in self.__messenger_token, one will be fetched from the client account data and stored

Returns messenger_token

Return type str

unread_notifications
Get all unread notifications (notifications that have not been recieved from a GET) and return them in a list

Returns unread notifications

Return type list<Notification>

home
generator for a client's subscriptions feed (home feed). Each iteration will return the next home post, in decending order of date posted

Returns generator iterating the home feed

Return type generator<Post>

smiles

Returns generator iterating posts that this client has smiled

Return type generator<Post>

comments

Returns generator iterating comments that this client has left

Return type generator<Comment>

next_req_id
Generate a new (sequential) sendbird websocket req_id in a thread safe way

Returns req_id

Return type str

user

Returns this client's user object

Return type *User*

unread_notifications_count

Returns number of unread notifications

Return type int

nick

Returns this client's username (`nick` name)

Return type str

email

Returns this client's associated email

Return type str

id

Returns this client's unique id

Return type str

fresh
Sets the update flag for this client, and returns it. Useful for when new information is pertanent

Returns self

Return type *Client*

achievements

Returns generator iterating this clients achievements

Return type generator<Achievement>

timeline
Alias for `self.user.timeline`

chats
generator for a Client's chats. Each iteration will return the next chat, in order of last message

Returns generator iterating through chats

Return type generator<Chat>

2.5 User

class ifunny.objects.**User**(**args*, ***kwargs*)
iFunny User object.

Parameters

- **id** (*str*) – id of the user
- **client** (Client) – Client that the user belongs to
- **data** (*dict*) – A data payload for the user to pull from before requests
- **paginated_size** (*int*) – number of items to get for each paginated request. If above the call type's maximum, that will be used instead

classmethod by_nick(*nick*, *client=<ifunny.objects._mixin.ClientBase object>*, ***kwargs*)
Get a user from their nick.

Parameters

- **nick** (*str*) – nick of the user to query. If this user does not exist, nothing will be returned
- **client** (Client) – the Client to bind the returned user object to

Returns A User with a given nick, if they exist

Return type *User*, or None

subscribe()
Subscribe to a user

Returns self

Return type *User*

unsubscribe()
Unsubscribe from a user

Returns self

Return type *User*

block(*type='user'*)
Block a user, either by account or device.

Parameters **type** (*str*) – Type of block. user blocks a user, installation blocks all users tied to a device

Returns self

Return type *User*

unblock()
Unblock a user.

Returns self

Return type *User*

report(*type*)
Report a user.

Parameters **type** (*str*) – Reason for report

hate -> hate speech

nude -> nudity

spam -> spam posting

target -> targeted harrassment

harm -> encouraging harm or violence

Returns self

Return type *User*

subscribe_to_updates()
Subscribe to update notifications from this User.

Returns self

Return type *User*

unsubscribe_to_updates()
Unsubscribe to update notifications from this User.

Returns self

Return type *User*

set_nick(*value*)
Change the nick of this User. This user must be you

Parameters **value** (*str*) – what to change the nick to

Returns self

Return type *User*

set_private(*value*)
Change the privacy value of this User This user must be you

Parameters **value** (*bool*) – set this user to private?

Returns self

Return type *User*

set_about(*value*)
Change the about of this User. This user must be you

Parameters **value** (*str*) – what to change the about to

Returns self

Return type *User*

timeline

Returns generator iterating user posts

Return type generator<Post>

subscribers

Returns generator iterating user subscipbers

Return type generator<User>

subscriptions

Returns generator iterating user subscriptions

Return type generator<User>

original_nick

Returns this users original nickname, if available

Return type string

post_count

Returns this users post count

Return type int

feature_count

Returns this users feature count

Return type int

smiles_count

Returns this users smile count

Return type int

subscriber_count

Returns this users subscriber count

Return type int

subscription_count

Returns this users subscruption count

Return type int

is_verified

Returns is this user verified?

Return type bool

is_banned

Returns is this user banned?

Return type bool

is_deleted

Returns is this user deleted?

Return type bool

days

Returns this users active days count

Return type int

rank

Returns this users meme experience rank

Return type str

nick_color

Returns this users nickname color

Return type str

chat_privacy

Returns this users chat privacy settings (privacy, public, subscribers)

Return type str

profile_image

Returns this accounts profile image, if any

Return type *Image*, or None

cover_image

Returns this accounts cover image, if any

Return type *Image*, or None

rating

Returns rating of this user with level data

Return type *Rating*

is_private

Returns is this profile private?

Return type bool

nick

Returns this users nickname

Return type str

about

Returns this users about section

Return type str

bans

Returns this users bans

Return type generator<Ban>

chat_url

Returns this users chat url, if `user.can_chat`

Return type str

chat

Returns this users chat, if `user.can_chat`

Return type *Chat*

is_blocked

Returns is this user blocked by me?

Return type bool

is_blocking_me

Returns is this user blocking me?

Return type bool

can_chat

Returns can I chat with this user?

Return type bool

is_updates_subscription

Returns am I subscribed to updates from this user?

Return type bool

is_subscribed

Returns is this user subscribed to me?

Return type bool

is_subscription

Returns am I subscribed to this user?

Return type bool

2.6 Post

class ifunny.objects.**Post**(**args*, ***kwargs*)

iFunny Post object

Parameters

- **id** (*str*) – id of the post
- **client** (Client) – Client that the post belongs to
- **data** (*dict*) – A data payload for the post to pull from before requests
- **paginated_size** (*int*) – number of items to get for each paginated request. If above the call type's maximum, that will be used instead

add_comment(*text=None*, *post=None*, *user_mentions=None*)

Add a comment to a post. At least one of the parameters must be used, as users shoud not post empty comments.

Parameters

- **text** (*str*) – Text of the comment, if any
- **post** (Post *or* *str*) – Post to post in the comment, if any. Can be a post id or a Post object, but the Post in reference must belong to the client creating the comment
- **user_mentions** (*list<User>*) – Users to mention, if any. Mentioned users must have their nick in the comment, and will be mentioned at the first occurance of their nick

Returns the posted comment

Return type *Comment*

smile()
smile a post. If already smiled, nothing will happen.

Returns self

Return type *Post*

remove_smile()
Remove a smile from a post. If none exists, nothing will happen.

Returns self

Return type *Post*

unsmile()
Unsmile a post. If already unsmiled, nothing will happen.

Returns self

Return type *Post*

remove_unsmile()
Remove an unsmile from a post. If none exists, nothing will happen.

Returns self

Return type *Post*

republish()
Republish this post. If this post is already republished by the client, nothing will happen.

Returns republished instance of this post, or None if already republished

Return type *Post*, or None

remove_republish()
Un-republish this post. This should work on an instance of this post from any User. If this post is not republished, nothing will happen.

Returns self

Return type *Post*

report(*type*)
Report a post.

Parameters **type** (*str*) – Reason for report

hate -> hate speech

nude -> nudity

spam -> spam posting

target -> targeted harrassment

harm -> encouraging harm or violence

Returns self

Return type *Post*

set_tags(*tags*)
Set the tags on your own post. If the post is not owned by the client, NotOwnContent exception is raised. Tags cannot include space characters, so those will be replace dropped.

Parameters **tags** (*list<str>*) – list of tags to add to set

Returns self

Return type *Post*

Raises NotOwnContent

delete()
Delete a post owned by the Client.

Returns self

Return type *Post*

pin()
Pin a post to the client user. Note that trying to pin a pinned post will return a `403`.

Returns self

Return type *Post*

unpin()
Unpin a post to the client user.

Returns self

Return type *Post*

set_schedule(*schedule*)
Update a delated posts scheduled time If post is not delated, nothing will happen

Parameters **schedule** (*int*) – new timestamp to be posted at

Returns self

Return type *Post*

set_visibility(*visibility*)
Update a delated posts visibility If post is not delated, nothing will happen

Parameters **visibility** (*str*) – visibility type. Can be one of (`public`, `subscribers`)

Returns self

Return type *Post*

read()
Mark this meme as read

Returns was this marked as read?

Return type bool

smiles

Returns generator iterating post smiles

Return type generator<User>

comments

Returns generator iterating post comments

Return type generator<Comment>

smile_count

Returns post's smile count

Return type int

unsmile_count

Returns post's unsmile count

Return type int

guest_smile_count

Returns post's smile count by guests

Return type int

comment_count

Returns post's comment count

Return type int

view_count

Returns post's view count

Return type int

republication_count

Returns post's republication count

Return type int

share_count

Returns post's share count

Return type int

author

Returns post's author

Return type *User*

source

Returns post's instance on it's original account, if a republication

Return type *Post*

is_original

Returns it this post original?

Return type bool

is_featured

Returns has this post been featured?

Return type bool

is_abused

Returns was this post removed by moderators?

Return type bool

type

Returns content ype of a post

Return type str

state

Returns the publicication state of the post

Return type str (published, ect)

boostable

Returns can this post be boosted?

Return type bool

created_at

Returns creation date timestamp

Return type int

published_at

Returns creation date timestamp

Return type int

content_url

Returns url pointing to the full sized image

Return type str

content

Returns image or video data from the post

Return type bytes

caption

Returns caption text for `caption` type posts

Return type str, or None

link

Returs this posts link

Return type str

is_republished

Returns is this post a republication?

Return type bool

is_smiled

Returns did I smile this post?

Return type bool

is_unsmiled

Returns did I unsmile this post?

Return type bool

visibility

Returns the visibility of a post

Return type str (public, subscribers, ect)

tags

Returns the tags of a post

Return type list<str>

is_pinned

Returns is this post pinned on it's authors profile?

Return type bool

2.7 Digest

class ifunny.objects.**Digest**(**args*, ***kwargs*)

iFunny digest object. represnets digests featured in explore, containing comments and posts

Parameters

- **id** (*str*) – id of the digest
- **client** (Client) – Client that the digest belongs to
- **data** (*dict*) – A data payload for the digest to pull from before requests
- **paginated_size** (*int*) – number of items to get for each paginated request. If above the call type's maximum, that will be used instead

read(*count=None*)

Mark posts in this digest as read. Will mark all read by default

Parameters **count** (*int*) – number of posts to mark as read

Returns self

Return type *Digest*

feed

Returns generator for posts that are in this digest

Return type generator<Post>

comments

Returns subscriber comments that are in this digest

Return type generator<Comment>

title

Returns the title of this digest

Return type str

smile_count

Returns number of smiles in this digest

Return type int

total_smiles

Returns alias for `Digest.smile_count`

Return type int

comment_count

Returns number of comments in this digest

Return type int

post_count

Returns number of posts in this digest

Return type int

unread_count

Returns number of unread posts in this digest

Return type int

count

Returns index of this digest

Return type int

index

Alias for `Digest.count`

2.8 Channel

class ifunny.objects.**Channel**(*id*, *client=<ifunny.objects._mixin.ClientBase object>*, *data={}*)

get(*key*, *default=None*)

feed

generator for a channels feed. Each iteration will return the next channel post, in decending order of date posted

Returns generator iterating the channel feed

Return type generator<Post>

2.9 Comment

class ifunny.objects.**Comment**(**args*, *post=None*, ***kwargs*)

iFunny Comment object

Parameters

- **id** (*str*) – id of the comment
- **client** (Client) – Client that the comment belongs to
- **data** (*dict*) – A data payload for the comment to pull from before requests
- **paginated_size** (*int*) – number of items to get for each paginated request. If above the call type's maximum, that will be used instead

reply(*text=''*, *post=None*, *user_mentions=None*)
Reply to a comment. At least one of the parameters must be used, as users cannot post empty replys.

Parameters

- **text** (*str*) – Text of the reply, if any
- **post** (*Post or str*) – Post to post in the reply, if any. Can be a post id or a Post object, but the Post in reference must belong to the client creating the reply
- **user_mentions** (*list<User>*) – Users to mention, if any. Mentioned users must have their nick in the reply, and will be mentioned at the first occurance of their nick

Raises RateLimit, TooManyMentions

Returns the posted reply

Return type *Comment*

delete()
Delete a comment

Raises RateLimit, BadAPIResponse

Returns self

Return type *Comment*

smile()
smile a comment. If already smiled, nothing will happen.

Returns self

Return type *Comment*

remove_smile()
Remove a smile from a comment. If none exists, nothing will happen.

Returns self

Return type *Comment*

unsmile()
Unsmile a comment. If already unsmiled, nothing will happen.

Returns self

Return type *Comment*

remove_unsmile()
Remove an unsmile from a comment. If none exists, nothing will happen.

Returns self

Return type *Comment*

report(*type*)
Report a comment.

Parameters **type** (*str*) – Reason for report

hate -> hate speech

nude -> nudity

spam -> spam posting

target -> targeted harrassment

harm -> encouraging harm or violence

Returns self

Return type *User*

replies

Returns generator iterating comment replies

Return type generator<Comment>

children

Returns generator iterating direct children of comments

Return type generator<Comment>

siblings

Returns generator iterating comment siblings

Return type generator<Comment>

content

Returns the text content of a comment

Return type str

cid

Returns the cid of this comment. A comments CID is the id of the post it's attached to

Return type str

state

Returns the state of the comment. Top comments are state top, and all others are state normal

Return type str (top, normal)

author

Returns the comment author

Return type Use

post

Returns the post that this comment is on

Return type *Post*

parent

Returns direct parent of this comment, or none for root comments

Return type *Comment*

root

Returns this comments root parent, or self if comment is root

Return type *Comment*

smile_count

Returns number of smiles on this comment

Return type int

unsmile_count

Returns number of unsmiles on this comment

Return type int

reply_count

Returns number of replies on this comment

Return type int

created_at

Returns creation date timestamp

Return type int

depth

Returns the depth of this comment

Return type int

is_root

Returns is this comment a top level (root) comment?

Return type bool

is_edited

Returns has this comment been deleted?

Return type bool

attached_post

Returns the attached post, if any

Return type *Post*, or None

user_mentions

Returns a list of mentioned users, if any

Return type list<User>

is_smiled

Returns did I smile this comment?

Return type bool

is_unsmiled

Returns did I unsmile this comment?

Return type bool

2.10 Chat

class ifunny.objects.**Chat**(**args*, ***kwargs*)

iFunny Chat object

Parameters

- **id** (*str*) – channel_url of the Chat. `Chat.channel_url` is aliased to this value, though `id` is more consistent with other mixin objects and how they update themselves.
- **client** (`Client`) – Client that the Chat belongs to
- **data** (*dict*) – A data payload for the Chat to pull from before requests
- **paginated_size** (*int*) – number of items to get for each paginated request. If above the call type's maximum, that will be used instead

classmethod by_link(*code*, *client=<ifunny.objects._mixin.ClientBase object>*, ***kwargs*)
Get a chat from it's code.

Parameters

- **code** (*str*) – code of the chat to query. If this user does not exist, nothing will be returned
- **client** (`Client`) – the Client to bind the returned user object to

Returns A Chat of the given code, if it exists

Return type *Chat*, or None

add_operator(*user*)
Add an operator toi a Chat

Params user operator to add

Returns fresh list of this chat's operators

Return type List<ChatUser>

remove_operator(*user*)
Remove an operator from a Chat

Params user operator to remove

Returns fresh list of this chat's operators

Return type List<ChatUser>

add_admin(*user*)
Add an administrator to this Chat

Parameters **user** (`User or ChatUser`) – the user that should be an admin

Returs self

Return type *Chat*

remove_admin(*user*)
Remove an administrator from this Chat

Parameters **user** (`User or ChatUser`) – the user that should no longer be an admin

Returs self

Return type *Chat*

join()
Join this chat

Returns did this client join successfuly?

Return type bool

leave()
Leave this chat

Returns did this client leave successfuly?

Return type bool

read()
Mark messages in a chat as read.

Returns self

Return type *Chat*

invite(*user*)
Invite a user or users to a chat.

Parameters **user** (*User, or list<User>*) – User or list<User> of invitees

Returs self

Return type *Chat*

kick(*user*)
Kick a member from a group

Parameters **user** (*User*) – User to kick

Returns self

Return type *Chat*

freeze(*until=0*, *callback=None*)
Freeze a Chat, and set the update flag.

Parameters

- **until** (*int*) – time in seconds to wait to unfreeze. If 0, there will be no unfreezing
- **callback** (*callable, or None*) – method to call when unfrozen, must accept single argument for Chat

Returs self

Return type *Chat*

unfreeze(*until=0*, *callback=None*)
Freeze a Chat, and set the update flag.

Parameters

- **until** (*int*) – time in seconds to wait to unfreeze. If 0, there will be no unfreezing
- **callback** (*callable, or None*) – method to call when unfrozen, must accept single argument for Chat

Returs self

Return type *Chat*

send_message(*message*, *read=False*)
Send a text message to a chat.

Parameters

- **message** (*str*) – text that you will send
- **read** (*bool*) – do we mark the chat as read?

Raises ChatNotActive if the attached client has not started the chat socket

Returns self

Return type *Chat*

send_image_url(*image_url*, *width=780*, *height=780*, *read=False*)
Send an image to a chat from a url source.

Parameters

- **image_url** (*str*) – url where the image is located. This should point to the image itself, not a webpage with an image
- **width** (*int*) – width of the image in pixels
- **height** (*int*) – heigh of the image in pixels
- **read** (*bool*) – do we mark the chat as read?

Raises ChatNotActive if the attached client has not started the chat socket

Returns self

Return type *Chat*

members

Returns generator to iterate through chat members

Return type generator<ChatUser>

messages

Returns generator to iterate through chat messages

Return type generator<Message>

send

Returns this classes send_message method

Return type function

admins

Returns list of chat admins, if group

Return type List<ChatUser>

operators

Returns list of chat operators, if group

Return type List<ChatUser>

title

Returns the title of this chat

Return type str

name
Alias for Chat.title

created

Returns timestamp of this chats creation data

Return type int

description

Returns admin defined description of the chat, if group

Return type str, or None

is_frozen

Returns is this chat frozen? Assumes False if attribute cannot be queried

Return type bool

type

Returns the type of this group. Can be `group`, `opengroup`, `chat`

Return type str

is_direct

Returns is this chat a private message chat?

Return type bool

is_private

Returns is this chat a private group?

Return type bool

is_public

Returns is this chat a public group?

Return type bool

member_count

Returs number of members in this chat

Return type int

muted

Returns is this chat muted by the client?

Return type bool

user

Returns This clients ChatUser in this chat

Return type *ChatUser*

2.11 ChatUser

class ifunny.objects.**ChatUser**(*id*, *chat*, **args*, *client=<ifunny.objects._mixin.ClientBase object>*, *sb_data=None*, ***kwargs*)

A User attatched to a chat. takes the same params as a User, with an extra set

Parameters

- **chat** (`Chat`) – Chat that this user is in
- **sb_data** (*dict*) – A sendbird data payload for the user to pull from before requests

kick()

Kick this member from a group

Returns self

Return type *ChatUser*

state

Returns Is this member invited (pending join), or joined?

Return type str

last_online

Returns timestamp of whne this user was last online

Return type int

online

Returns is this user online?

Return type bool

chat

Returns this users chat, if `user.can_chat`

Return type *Chat*

2.12 Message

class ifunny.objects.**Message**(*id*, *channel_url*, *client*, *data=None*)

Sendbird message object. Created when a message is recieved.

Parameters

- **data** (*dict*) – message json, data after prefix in a sendbird websocket response
- **client** (`Client`) – client that the object belongs to

delete()

Delete a message sent by the client. This is exparamental, and may not work

Returns self

Return type *Message*

author

Returns the author of this message

Return type *ChatUser*

chat

Returns Chat that this message exists in

Return type *Chat*

content

Returns String content of the message

Return type str

channel_url

Returns chat url for this messages chat

Return type str

send

Returns the send() method of this messages chat for easy replies

Return type function

send_image_url

Returns the send_image_url() method of this messages chat for easy replies

Return type function

type

Returns type of message. Text messages will return type MESG, while files return the file mime

Return type str

file_url

Returns message file url, if any

Return type str, or None

file_data

Returns file binary data, if any

Return type str, or None

file_type

Returns file type, if the message is a file

Return type str, or None

file_name

Returns file name, if the message is a file

Return type str, or None

2.13 ChatInvite

class ifunny.objects.**ChatInvite**(*data*, *client*)

Chat update class. Created when an invite is recieved from the chat websocket.

Parameters

- **data** (*dict*) – chat json, data after prefix in a sendbird websocket response
- **client** (Client) – client that the object belongs to

headers

accept()

Accept an incoming invitation, if it is from a user. If it is not, the method will do nothing and return None.

Returns Chat that was joined, or None

Return type *Chat*, or None

decline()

Decline an incoming invitation, if it is from a user. If it is not, the method will do nothing and return None.

url

Returns the request url to the incoming Chat

Return type str

channel_url

Returns the url to the incoming Chat

Return type str

chat

Returns the incoming Chat

Return type *Chat*

inviter

Returns if this update is an invite, returns the inviter

Return type *User*, or None

invitees

Returns if this update is an invite, returns the invitees

Return type list<User>, or None

type

Returns the type of the incoming chat data

Return type str

2.14 Image

class ifunny.objects.**Image**(*url*, *background=None*, *client=<ifunny.objects._mixin.ClientBase object>*)

Wrapper for image properties

Parameters

- **url** (*str*) – location of the image
- **background** (*str*) – image background color
- **client** (Client) – client who requests the image

content

Returns image content

Return type bytes

2.15 Rating

class ifunny.objects.**Rating**(*user*, *client=<ifunny.objects._mixin.ClientBase object>*, *data=None*)

iFunny profile ratings

Parameters

- **user** (User) – user who this rating is of

- **client** (Client) – client who requests the rating
- **data** (*dict*) – data payload of this rating

get(*key*, *default=None*)

fresh

Returns this object with the update flag set

Return type *Rating*

points

Returns the points of this user

Return type int

visible

Returns is the level of this user visible?

Return type bool

level

Returns the level of this user

Return type int

level_points

Returns the points required for the level of this user

Return type int

next

Returns the next level of this user

Return type int

next_points

Returns the points required for the next level of this user

Return type int

max

Returns the max level of this user

Return type int

max_points

Returns the points required for the max level of this user

Return type int

2.16 Ban

class ifunny.objects.**Ban**(**args*, *user=None*, ***kwargs*)

iFunny ban subclass of ObjectMixin

get(*key*, *default=None*)

reason

Returns reason for this ban

Return type str

created_at

Returns timestamp of when this ban was created

Return type int

expires_at

Returns timestamp of when this ban expires

Return type int

type

Returns type of ban

Return type str

index

Returns ban index relative to other bans (starting at 1)

Return type int

is_appealed

Returns has this ban been appealed?

Return type bool

is_appealable

Returns can this ban be appealed?

Return type bool

was_shown

Returns was the client notified of this ban?

Return type bool

is_active

Returns is this ban active?

Return type bool

is_shortable

Returns can this ban be shortened?

Return type bool

2.17 Achievement

class ifunny.objects.**Achievement**(**args*, *user=None*, ***kwargs*)

iFunny achievements subclass of ObjectMixin

get(*key*, *default=None*)

tasks

Returns tasks to complete this achievement

Return type list<Task>

season

Returns this achievements season

Return type *Season*

start_at

Returns achievement start timestamp

Return type int

expire_at

Returns achievement expiration timestamp

Return type int

was_shown

Returns was this achievement shown?

Return type bool

title

Returns this achievements title

Return type str

description

Returns this achievements description

Return type str

action_text

Returns this achievements action_text

Return type str

period

Returns this achievements period

Return type str

status

Returns this achievements status

Return type str

reward

Returns this achievements reward

Return type int

complete_text

Returns this achievements text when complete

Return type str

complete_description

Returns this achievements description when complete

Return type str

2.18 Task

class ifunny.objects._small.**Task**(*id*, *achievement*, *data=None*)

Achievement task

Parameters

- **id** (*str*) – id of this task
- **achievement** (Achievement) – achievement that this task is for
- **data** (*dict*) – data payload of this task

get(*key*, *default=None*)

fresh

Returns this object with the update flag set

Return type *Task*

count

Returns times to complete task required

Return type int

event

Returns task event

Return type str

2.19 Season

class ifunny.objects._small.**Season**(*achievement*, *data=None*)

Achievement season

Parameters

- **achievement** (Achievement) – achievement that this season is for
- **data** (*dict*) – data payload of this task

get(*key*, *default=None*)

id

Returns the id of this season

Return type str

title

Returns the title of this season

Return type str

description

Returns the description of this season

Return type str

status

Returns the status of this season

Return type str

start_at

Returns the start_at timestamp

Return type int

expire_at

Returns the expire_at timestamp

Return type int

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Index

D

E

F

G

H

I

J

K

T

U

V

W