discord.aio Documentation

Release 0.3.1

Ryozuki

Dec 14, 2020

Contents:

See the *reference documentation*.

CHAPTER 1

discord.aio

discord.aio is an asynchronous Discord API wrapper

Currently under very early development

Python 3.6+ only.

1.1 Documentation

You can find the module documentation here: documentation

1.2 Installation

1.2.1 With pip:

- `pip3 install discord.aio`

1.2.2 From source:

- `git clone https://github.com/Ryozuki/discord.aio && cd discord.aio && pip3 install .`

1.3 Local development

- `git clone https://github.com/Ryozuki/discord.aio`
- `cd discord.aio && pip3 install -e .`

1.4 Example bot

```
import asyncio
import os
import logging
from discordaio import DiscordBot

logging.basicConfig(
    level='DEBUG', format='%(asctime)s - %(name)s - %(levelname)s: %(message)s')
logger = logging.getLogger('my_lovely_bot')

if __name__ == '__main__':
    TOKEN = os.environ['DISCORD_TOKEN']

    bot = DiscordBot(TOKEN)

    @bot.event()
    async def on_ready():
        logger.info('Connected!')
        logger.info(f'My username is {bot.user}')

    @bot.event('on_message') # You can also use a custom function name.
    async def foo_bar(message):
        logger.info(f'{message.author}: {message.content}')

    bot.run()
```

Here you can find a more extensive example.

1.5 TODO

- Add compression support
- Add bot shards support
- Handle ISO8601 timestamp
- Make the DiscordBot methods better.

1.5.1 Your first bot

TODO: Do this

1.5.2 Events

Events

Event: on_ready

Raised when:

- The client is ready and connected.

Note: Before this event is raised, DiscordBot.user is filled with information.

```
@bot.event()
async def on_ready():
    print('Connected!')
    print(f'My username is {bot.user}')
```

Event: on_resumed

Raised when:

- A client has sent a resume payload to the gateway (for resuming existing sessions).

Event: on_invalid_session

Raised when:

- The gateway could not initialize a session after receiving an Opcode 2 Identify
- The gateway could not resume a previous session after receiving an Opcode 6 Resume
- The gateway has invalidated an active session and is requesting client action

Event Parameters:

- `resume` (**bool**): Indicates whether the client can resume.

Event: on_channel_create

Raised when:

- A new channel is created

Event Parameters:

- `channel` (**Channel**): The created channel

Event: on_channel_update

Raised when:

- A channel is updated

Event Parameters:

- `channel` (**Channel**): The updated channel

Event: on_channel_delete

Raised when:

- A channel is deleted

Event Parameters:

- `channel` (**Channel**): The deleted channel

Event: on_channel_pin

Raised when:

- A message is pinned or unpinned in a text channel.

Note: This is not raised when a pinned message is deleted.

Event Parameters:

- `channel_id` (**int**): The id of the channel
- `last_pin_timestamp` (**int**): The time at which the most recent pinned message was pinned

Event: on_guild_create

Raised when:

- After the `on_ready` event, to fullfill the guild information.
- When a Guild becomes available again to the client.
- When the current user joins a new Guild.

Event Parameters:

- `guild` (**Guild**): The guild

```
@bot.event()
async def on_guild_create(guild):
    print(f'I\'m connected to {guild.name} guild, it got {len(guild.channels)}␣
↪channels.')
```

Event: on_guild_delete

Raised when:

- A guild becomes unavailable during a guild outage
- The user leaves or is removed from a guild
- When the current user joins a new Guild.

Note: If the unavailable attribute is not set, the user was removed from the guild.

Event Parameters:

- `guild` (**Guild**): The guild

```
@bot.event()
async def on_guild_delete(guild):
    print(f'{guild.name} went offline?')
    if not guild.unavailable:
        print(f'I got removed from {guild}!')
```

Event: on_guild_emojis_update

Raised when:

- When a guild's emojis have been updated.

Event Parameters:

- `guild_id` (**int**): The guild id
- `emojis` (**list**): A list of emojis

Event: on_guild_integrations_update

Raised when:

- When a guild integration is updated.

Event Parameters:

- `guild_id` (**int**): The guild id.

Event: on_guild_member_add

Raised when:

- When a new user joins a guild.

Event Parameters:

- `guild_id` (**int**): The guild id.
- `member` (**GuildMember**): The user that joined.

Event: on_guild_member_remove

Raised when:

- A user is removed from a guild (leave/kick/ban).

Event Parameters:

- `guild_id` (**int**): The guild id.
- `user` (**User**): The user that was removed/left.

Event: on_guild_member_update

Raised when:

- A guild member is updated

Event Parameters:

- `guild_id` (**int**): The guild id.
- `roles` (**list**): User role ids.
- `user` (**User**): The user.
- `nick` (**str**): Nickname of the user in the guild.

Event: on_guild_members_chunk

Raised when:

- In response to Guild Request Members.

Event Parameters:

- `guild_id` (**int**): The guild id.
- `members` (**list**): Set of guild members

Event: on_guild_role_create

Raised when:

- A guild role is created.

Event Parameters:

- `guild_id` (**int**): The guild id.
- `role` (**Role**): The role created

Event: on_guild_role_update

Raised when:

- A guild role is updated.

Event Parameters:

- `guild_id` (**int**): The guild id.
- `role` (**Role**): The role updated

Event: on_guild_role_delete

Raised when:

- A guild role is deleted.

Event Parameters:

- `guild_id` (**int**): The guild id.
- `role_id` (**int**): The id of the deleted role

Event: on_ban

Raised when:

- A user is banned from a guild

Event Parameters:

- `guild_id` (**int**): The guild id.
- `user` (**User**): The banned .

Event: on_ban_remove

Raised when:

- A user is unbanned from a guild

Event Parameters:

- `guild_id` (**int**): The guild id.
- `user` (**User**): The unbanned user.

Event: on_message

Raised when:

- A user send a message to a channel

Event Parameters:

- `user_id` (**int**): The id of the user that started typing.
- `channel_id` (**int**): The id of the channel where the action happened.
- `timestamp` (**int**): The timestamp telling when it happened.

```
@bot.event()
async def on_message(message):
    print(f'{message.author}: {message.content}')
```

Event: on_message_update

Raised when:

- A message is updated.

Note: Unlike creates, message updates may contain only a subset of the full message object payload (but will always contain an id and channel_id)

Event Parameters:

- `message` (**ChannelMessage**): The Channel message that has been updated

```
@bot.event()
async def on_message_create(message):
    print(f'A message with id {message.id} has been updated.')
```

Event: on_message_delete

Raised when:

- A message is deleted.

Event Parameters:

- `id` (**int**): The id of the message.
- `channel_id` (**int**): The id of the channel.

```
@bot.event()
async def on_message_delete(id, channel_id):
    print(f'A message with id {id} has been deleted.')
```

Event: on_message_delete_bulk

Raised when:

- Multiple messages are deleted at once.

Event Parameters:

- `ids` (**int**): The ids of the messages
- `channel_id` (**int**): The id of the channel

```
@bot.event()
async def on_message_delete_bulk(ids, channel_id):
    print(f'Multiple messages have been deleted')
```

Event: on_message_reaction_add

Raised when:

- A user adds a reaction to a message

Event Parameters:

- `user_id` (**int**): The id of the user
- `channel_id` (**int**): The id of the channel
- `message_id` (**int**): The id of the message
- `emoji` (**Emoji**): The emoji used to react

```
@bot.event()
async def on_message_reaction_add(user_id, channel_id, message_id, emoji):
    user = await bot.get_user(user_id)
    print(f'{user} reacted to a message with {emoji.name}')
```

Event: on_message_reaction_remove

Raised when:

- A user removes a reaction from a message

Event Parameters:

- `user_id` (**int**): The id of the user
- `channel_id` (**int**): The id of the channel
- `message_id` (**int**): The id of the message
- `emoji` (**Emoji**): The emoji used to react

```
@bot.event()
async def on_message_reaction_add(user_id, channel_id, message_id, emoji):
    user = await bot.get_user(user_id)
    print(f'{user} removed reaction {emoji.name} from a message.')
```

Event: on_message_reaction_remove_all

Raised when:

- A user explicitly removes all reactions from a message.

Event Parameters:

- `channel_id` (**int**): The id of the channel
- `message_id` (**int**): The id of the message

Event: on_presence_update

Raised when:

- A user's presence is updated for a guild.

Note: The user object within this event can be partial, the only field which must be set is the id field

Event Parameters:

- `user` (**User**): The user presence is being updated for
- `roles` (**list**): Ids of the roles this user is in
- `game` (**?Activity**): Null, or the user's current activity
- `guild_id` (**int**): The guild id
- `status` (**str**): Either "idle", "dnd", "online", or "offline"

Event: on_typing_start

Raised when:

- A user starts typing in a channel

Event Parameters:

- `user_id` (**int**): The id of the user that started typing.
- `channel_id` (**int**): The id of the channel where the action happened.
- `timestamp` (**int**): The timestamp telling when it happened.

```
@bot.event()
async def on_typing_start(user_id, channel_id, timestamp):
    user = await bot.get_user(user_id)
    print(f'{user} started typing!')
```

Event: on_user_update

Raised when:

- Properties about the user change

Event Parameters:

- `user` (**User**): The user that has been updated

Event: on_voice_state_update

Raised when:

- Someone joins/leaves/moves voice channels.

Event Parameters:

- `voice_state` (**VoiceState**): The voice state object

Event: on_voice_server_update

Raised when:

- A guild's voice server is updated.
- This is sent when initially connecting to voice, and when the current voice instance fails over to a new server.

Event Parameters:

- `token` (**str**): Voice connection token-
- `guild_id` (**int**): The guild this voice server update is for
- `endpoint` (**str**): The voice server host

Event: on_webhooks_update

Raised when:

- A guild's voice server is updated.
- This is sent when initially connecting to voice, and when the current voice instance fails over to a new server.

Event Parameters:

- `guild_id` (**int**): Id of the guild
- `channel_id` (**int**): Id of the channel

1.5.3 discordaio

discord.aio is an asynchronous Discord API wrapper for python 3.6+

Submodules

discordaio.activity

Classes

- *Activity*: Represents a discord activity
- *ActivityParty*: Activity party
- *ActivityTimestamps*: Activity timestamps
- *ActivityAssets*: Activity assets

class discordaio.activity.**Activity**(*name=''*, *type=0*, *url=''*, *timestamps=None*, *application_id=0*, *details=''*, *state=''*, *party=None*, *assets=None*)

Represents a discord activity

New in version 0.2.0.

name
the activity's name

Type str

type
activity type

Type int

url
stream url, is validated when type is 1

Type str, optional

timestamps
object unix timestamps for start and/or end of the game

Type Timestamps

application_id
application id for the game

Type int, optional

details
what the player is currently doing

Type str, optional

state
the user's current party status

Type str, optional

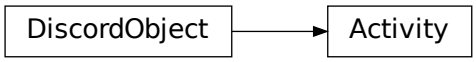
party
object information for the current party of the player

Type Party

assets
object images for the presence and their hover texts

Type Assets

Inheritance



class discordaio.activity.**ActivityParty**(*id=''*, *size=[]*)
Activity party

New in version 0.2.0.

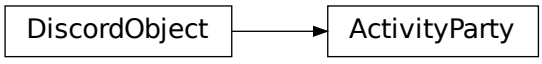
id
the id of the party

Type str, optional

size
array of two integers (current_size, max_size), used to show the party's current and maximum size

Type list of int

Inheritance



class discordaio.activity.**ActivityTimestamps**(*start=0*, *end=0*)
Activity timestamps

New in version 0.2.0.

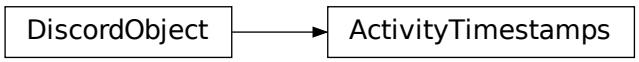
start
unix time (in milliseconds) of when the activity started

Type int, optional

end
unix time (in milliseconds) of when the activity ends

Type int, optional

Inheritance



class discordaio.activity.**ActivityAssets**(*large_image=''*, *large_text=''*, *small_image=''*, *small_text=''*)

Activity assets

New in version 0.2.0.

large_image
the id for a large asset of the activity, usually a snowflake

Type str, optional

large_text
text displayed when hovering over the large image of the activity

Type str, optional

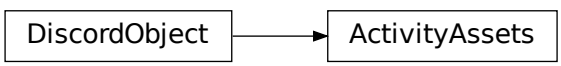
small_image
the id for a small asset of the activity, usually a snowflake

Type str, optional

small_text
text displayed when hovering over the small image of the activity

Type str, optional

Inheritance



discordaio.base

Classes

- *DiscordObject*: Base class for discord objects.

class discordaio.base.**DiscordObject**
Base class for discord objects.

Inheritance

classmethod await from_api_res(*coro_or_json_or_str*, *bot: Optional[discordaio.client.DiscordBot] = None*)
Parses a discord API response

discordaio.channel

Contains all related Channel Discord objects

Classes

- *Channel*: Represents a guild or DM channel within Discord.
- *ChannelMessage*: Represents a message sent in a channel within Discord.
- *Overwrite*: Represents a Overwrite object.
- *MessageActivity*: Represents a Message Activity.
- *MessageApplication*: Represents a Message Application.
- *Reaction*: Represents a Reaction.
- *Embed*: Represents a discord Embed
- *EmbedThumbnail*: Represents a embed thumbnail object

- *EmbedVideo*: Represents a embed video
- *EmbedImage*: Represents a embed image
- *EmbedProvider*: Represents a embed provider
- *EmbedAuthor*: Represents a embed author
- *EmbedFooter*: Represents a embed footer
- *EmbedField*: Represents a embed field
- *Attachment*: Represents a attachment

class discordaio.channel.**Channel**(*id: int = None*, *type: int = 0*, *guild_id: int = None*, *position: int = None*, *permission_overwrites: List[discordaio.channel.Overwrite] = []*, *name: str = None*, *topic: str = None*, *nsfw: bool = False*, *last_message_id: int = None*, *bitrate: int = None*, *user_limit: int = None*, *recipients: List[discordaio.user.User] = []*, *icon: str = None*, *owner_id: int = None*, *application_id: int = None*, *parent_id: int = None*, *last_pin_timestamp: int = None*)

Represents a guild or DM channel within Discord.

New in version 0.2.0.

id
the id of this channel

type
the value_type of channel

guild_id
the id of the guild

position
sorting position of the channel

permission_overwrites
explicit permission overwrites for members and roles

name
the name of the channel (2-100 characters)

topic the channel topic

Type 0-1024 characters

nsfw
if the channel is nsfw

last_message_id
the id of the last message sent in this channel (may not point to an existing or valid message)

bitrate
the bitrate (in bits) of the voice channel

user_limit
the user limit of the voice channel

recipients
the recipients of the DM

icon
icon hash

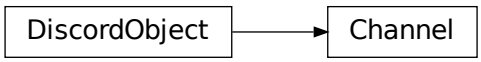
owner_id
id of the DM creator

application_id application id of the group DM creator if it is bot-created

parent_id
id of the parent category for a channel

last_pin_timestamp
timestamp when the last pinned message was pinned

Inheritance



await bulk_delete_messages(*ids: List[T]*)
Delete multiple messages in a single request. This endpoint can only be used on guild channels and requires the MANAGE_MESSAGES permission.

This endpoint will not delete messages older than 2 weeks, and will fail if any message provided is older than that.

New in version 0.3.0.

await create_invite(*max_age: int*, *max_uses=0*, *temporary=False*, *unique=False*) → discordaio.invite.Invite
Create a new invite object for the channel. Only usable for guild channels. Requires the CREATE_INSTANT_INVITE permission.

New in version 0.3.0.

await delete()
Deletes the channel.

New in version 0.3.0.

await delete_permission(*overwrite_id*)
Delete a channel permission overwrite for a user or role in a channel. Only usable for guild channels. Requires the MANAGE_ROLES permission.

New in version 0.3.0.

await delete_pinned_message(*message_id*)
Delete a pinned message. Requires the MANAGE_MESSAGES permission.

New in version 0.3.0.

await get_invites() → List[discordaio.invite.Invite]
Returns a list of invite objects (with invite metadata) for the channel. Only usable for guild channels. Requires the MANAGE_CHANNELS permission.

New in version 0.3.0.

await get_message(*message_id*) → discordaio.channel.ChannelMessage
Gets a specific channel message.

await get_messages(*limit: int = None*, *around: int = None*, *before: int = None*, *after: int = None*) → List[discordaio.channel.ChannelMessage]
Gets channel messages

New in version 0.3.0.

await get_pinned_messages() → List[discordaio.channel.ChannelMessage]
Returns all pinned messages in the channel as an array of message objects.

New in version 0.3.0.

mention() → str
Returns formatted channel mention.

New in version 0.3.0.

await pin_message(*message_id*)
Pin a message in a channel. Requires the MANAGE_MESSAGES permission.

New in version 0.3.0.

await refresh() → discordaio.channel.Channel
Returns a "refreshed" channel instance, may be used to make sure you have the latest state of the channel.

New in version 0.3.0.

Example

channel = await channel.refresh()

Returns The requested channel.

await send_message(*msg: str*, *tts=False*)
New in version 0.3.0.

Parameters

- **msg** – The message to send. If it's over 2000 chars it will be splitted.
- **tts** – Wether to send it as a tts message.

TODO: Add embed and files.

await typing()
Start typing.

New in version 0.3.0.

await update() → discordaio.channel.Channel
Updates the channel using the current channel instance properties.

New in version 0.3.0.

Returns The updated channel.

class discordaio.channel.**ChannelMessage**(*id=None*, *channel_id=None*, *author: discordaio.user.User = None*, *content: str = None*, *timestamp: int = None*, *edited_timestamp: int = None*, *tts: bool = False*, *mention_everyone: bool = False*, *mentions: List[discordaio.user.User] = []*, *mention_roles: List[discordaio.role.Role] = []*, *attachments: List[discordaio.channel.Attachment] = []*, *embeds: List[discordaio.channel.Embed] = []*, *reactions: List[discordaio.channel.Reaction] = []*, *nonce: int = None*, *pinned=False*, *webhook_id=None*, *type=None*, *activity=<discordaio.channel.MessageActivity object>*, *application=<discordaio.channel.MessageApplication object>*)

Represents a message sent in a channel within Discord.

New in version 0.2.0.

Note: The author object follows the structure of the *User* object, but is only a valid user in the case where the message is generated by a user or bot user. If the message is generated by a *Webhook*, the author object corresponds to the webhook's id, username, and avatar. You can tell if a message is generated by a webhook by checking for the webhook_id on the message object.

---

id
id of the message

channel_id
id of the channel the message was sent in

author
the author of this message (not guaranteed to be a valid user, see below)

content
contents of the message

timestamp
timestamp when this message was sent

edited_timestamp
timestamp when this message was edited (or null if never)

tts
whether this was a TTS message

mention_everyone
whether this message mentions everyone

mentions
objects users specifically mentioned in the message

mention_roles
object ids roles specifically mentioned in this message

attachments
objects any attached files

embeds
objects any embedded content

reactions
objects reactions to the message

nonce
used for validating a message was sent

pinned whether this message is pinned

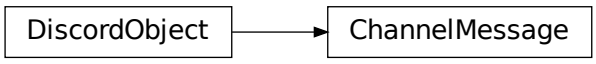
webhook_id
if the message is generated by a webhook, this is the webhook's id

type
type of message

activity
activity object sent with Rich Presence-related chat embeds

application
application object sent with Rich Presence-related chat embeds

Inheritance



await delete()
Delete a message. If operating on a guild channel and trying to delete a message that was not sent by the current user, this endpoint requires the MANANGE_MESSAGES permission

await delete_all_reactions()
Deletes all reactions on a message. This endpoint requires the 'MANAGE_MESSAGES' permission to be present on the current user.

await delete_own_reaction(*emoji_id*)
Delete a reaction the bot has made for the message

await delete_user_reaction(*user_id*, *emoji_id*)
Deletes another user's reaction. This endpoint requires the 'MANAGE_MESSAGES' permission to be present on the current user.

await get_reactions(*emoji_id*, *before: int = None*, *after: int = None*, *limit: int = None*) → List[discordaio.user.User]
Get a list of users that reacted with this emoji.

await update() → discordaio.channel.ChannelMessage
Updates a previously sent message with the current instance properties (content). You can only edit messages that have been sent by the current user. Returns a message object. Fires a Message Update Gateway event.

TODO: Add embed support

class discordaio.channel.**Overwrite**(*id=None*, *type=None*, *allow=None*, *deny=None*)
Represents a Overwrite object.

New in version 0.2.0.

id
role or user id

Type int

type
either “role” or “member”

Type str

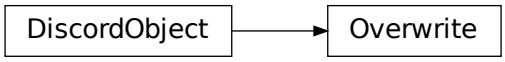
allow
permission bit set

Type int

deny
permission bit set

Type int

Inheritance



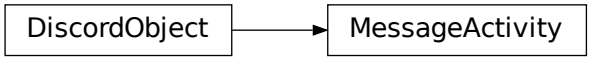
class discordaio.channel.**MessageActivity**(*type: int = None*, *party_id: int = None*)
Represents a Message Activity.

New in version 0.2.0.

type
type of message activity

party_id
party_id from a Rich Presence event

Inheritance



class discordaio.channel.**MessageApplication**(*id=None*, *cover_image=None*, *description=None*, *icon=None*, *name=None*)
Represents a Message Application.

New in version 0.2.0.

id
id of the application

Type int

cover_image
id of the embed's image asset

Type str

description
application's description

Type str

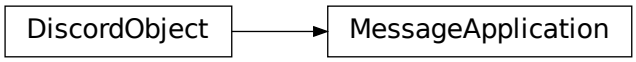
icon
id of the application's icon

Type str

name
name of the application

Type str

Inheritance



class discordaio.channel.**Reaction**(*count: int = None*, *me: bool = False*, *emoji: discordaio.emoji.Emoji = None*)

Represents a Reaction.

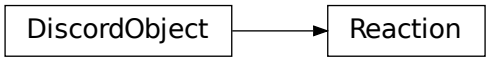
New in version 0.2.0.

count
times this emoji has been used to react

me
whether the current user reacted using this emoji

emoji emoji information

Inheritance



class discordaio.channel.**Embed**(*title=None*, *type=None*, *description=None*, *url=None*, *timestamp=None*, *color=None*, *footer=<discordaio.channel.EmbedFooter object>*, *image=<discordaio.channel.EmbedImage object>*, *thumbnail=<discordaio.channel.EmbedThumbnail object>*, *video=<discordaio.channel.EmbedVideo object>*, *provider=<discordaio.channel.EmbedProvider object>*, *author=<discordaio.channel.EmbedAuthor object>*, *fields=[]*)

Represents a discord Embed

New in version 0.2.0.

title
title of embed

Type str

type
type of embed (always "rich" for webhook embeds)

Type str

description
description of embed

Type str

url
url of embed

Type str

timestamp
timestamp of embed content

Type int

color
color code of the embed

Type int

footer
footer information

Type *EmbedFooter*

image
image information

Type *EmbedImage*

thumbnail
thumbnail information

Type *EmbedThumbnail*

video
video information

Type *EmbedVideo*

provider
provider information

Type *EmbedProvider*

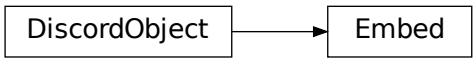
author
author information

Type *EmbedAuthor*

fields
fields information

Type list of *EmbedField*

Inheritance



class discordaio.channel.**EmbedThumbnail**(*url=None*, *proxy_url=None*, *height=None*, *width=None*)

Represents a embed thumbnail object

New in version 0.2.0.

url
source url of thumbnail (only supports http(s) and attachments)

Type str

proxy_url
a proxied url of the thumbnail

Type str

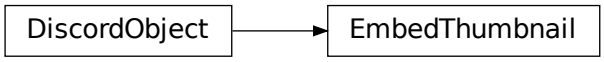
height
height of thumbnail

Type int

width
width of thumbnail

Type int

Inheritance



class discordaio.channel.**EmbedVideo**(*url=None*, *height=None*, *width=None*)

Represents a embed video

New in version 0.2.0.

url

source url of video

Type str

height

height of video

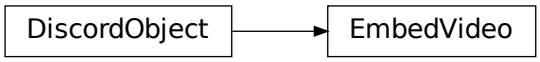
Type int

width

width of video

Type int

Inheritance



class discordaio.channel.**EmbedImage**(*url=None*, *proxy_url=None*, *height=None*, *width=None*)

Represents a embed image

New in version 0.2.0.

url

source url of image (only supports http(s) and attachments)

Type str

proxy_url

a proxied url of the image

Type str

height

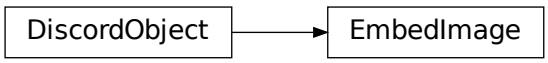
height of image

Type int

width
width of image

Type int

Inheritance



class discordaio.channel.**EmbedProvider**(*name=None*, *url=None*)
Represents a embed provider

New in version 0.2.0.

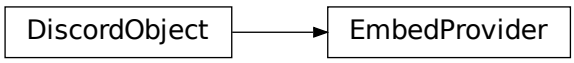
name
name of provider

Type str

url
url of provider

Type str

Inheritance



class discordaio.channel.**EmbedAuthor**(*name=None*, *url=None*, *icon_url=None*, *proxy_icon_url=None*)
Represents a embed author

New in version 0.2.0.

name
name of author

Type str

url
url of author

Type str

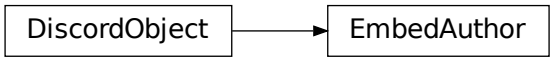
icon_url
url of author icon (only supports http(s) and attachments)

Type str

proxy_icon_url
a proxied url of author icon

Type str

Inheritance



class discordaio.channel.**EmbedFooter**(*text=None*, *icon_url=None*, *proxy_icon_url=None*)
Represents a embed footer

New in version 0.2.0.

text
footer text

Type str

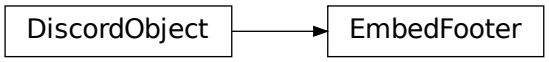
icon_url
url of footer icon (only supports http(s) and attachments)

Type str

proxy_icon_url
a proxied url of footer icon

Type str

Inheritance



class discordaio.channel.**EmbedField**(*name=None*, *value=None*, *inline=False*)
Represents a embed field

New in version 0.2.0.

name
name of the field

Type str

value
value of the field

Type str

inline
whether or not this field should display inline

Type bool

Inheritance

class discordaio.channel.**Attachment**(*id=None*, *filename=None*, *size=None*, *url=None*, *proxy_url=None*, *height=None*, *width=None*)

Represents a attachment

New in version 0.2.0.

id
attachment id

Type int

filename
name of file attached

Type str

size
size of file in bytes

Type int

url
source url of file

Type str

proxy_url
a proxied url of file

Type str

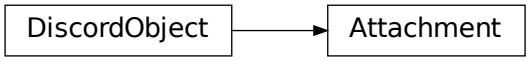
height
height of file (if image)

Type int

width
width of file (if image)

Type int

Inheritance



discordaio.client

- *Classes*

Classes

- *DiscordBot*: This class represents a discord bot object, it has many utility methods for making a bot.

class discordaio.client.**DiscordBot**(*token: str*)
This class represents a discord bot object, it has many utility methods for making a bot.

New in version 0.2.0.

token
The discord token used for authentication

Type str

http
Used for making http requests and websocket creation.

Type *HTTPHandler*

guilds
The list of guilds the bot is in.

Type list of *Guild*

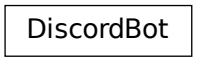
user
The user object of the bot.

Type *User*

ws
The websocket used for communication

Type *DiscordWebsocket*

Inheritance



event (*name: str = None*)
DiscordBot event decorator, uses the function's name or the 'name' parameter to subscribe to a event

New in version 0.2.0.

await exit ()
Disconnects the bot

New in version 0.2.0.

await get_channel (*channel_id: int*) → discordaio.channel.Channel
Gets a channel from it's id

New in version 0.2.0.

Parameters **channel_id** – The channel id

await get_dms () → List[discordaio.channel.Channel]
Gets a list of dms.

New in version 0.2.0.

Returns The DMs channels

Return type `list` of *Channel*

await get_guild (*guild_id: int*) → discordaio.guild.Guild
Returns a Guild object from a guild id.

New in version 0.2.0.

Parameters **guild_id** (`int`) – The guild id

Returns The requested guild

Return type *Guild*

await get_guilds () → List[discordaio.guild.Guild]
Returns a list of guilds where the bot is in.

New in version 0.2.0.

await get_self_user () → discordaio.user.User
Returns the bot user object. (it's like *get_user('@me')*)

New in version 0.2.0.

await get_user (*id*) → discordaio.user.User
Gets the user object from the given user id.

New in version 0.2.0.

Parameters **id** (`int`) – The user id

Returns The requested user

Return type *User*

await move_channels(*guild_id: int, array: List[Tuple[int, int]]*)
Modify the positions of a set of channel objects for the guild.

Note: Requires 'MANAGE_CHANNELS' permission. Fires multiple Channel Update events. Only channels to be modified are required, with the minimum being a swap between at least two channels.

Parameters

- **guild_id** – The id of the guild.
- **array** – A list of tuples containing (channel_id, position)

run() → None
Starts the bot, making it connect to discord.

New in version 0.2.0.

discordaio.constants

- *Variables*

Variables

- *DISCORD_CDN*
- *DISCORD_API_URL*

discordaio.constants.**DISCORD_CDN**
str(object='') -> str str(bytes_or_buffer[, encoding[, errors]]) -> str

Create a new string object from the given object. If encoding or errors is specified, then the object must expose a data buffer that will be decoded using the given encoding and error handler. Otherwise, returns the result of object.__str__() (if defined) or repr(object). encoding defaults to sys.getdefaultencoding(). errors defaults to 'strict'. .. code-block:: text

'https://cdn.discordapp.com'

discordaio.constants.**DISCORD_API_URL**
str(object='') -> str str(bytes_or_buffer[, encoding[, errors]]) -> str

Create a new string object from the given object. If encoding or errors is specified, then the object must expose a data buffer that will be decoded using the given encoding and error handler. Otherwise, returns the result of object.__str__() (if defined) or repr(object). encoding defaults to sys.getdefaultencoding(). errors defaults to 'strict'. .. code-block:: text

'https://discordapp.com/api/v8'

discordaio.emoji

Classes

- *Emoji*: Represents a emoji object

class discordaio.emoji.**Emoji**(*id=0*, *name=''*, *roles=[]*, *user=None*, *require_colons=False*, *managed=False*, *animated=False*)

Represents a emoji object

New in version 0.2.0.

id
emoji id

Type int

name
emoji name

Type str

roles
object ids roles this emoji is whitelisted to

Type list of *Role*

user
object user that created this emoji

Type *User*

require_colons
whether this emoji must be wrapped in colons

Type bool, optional

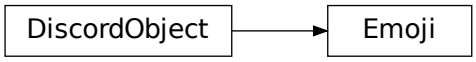
managed
whether this emoji is managed

Type bool, optional

animated
whether this emoji is animated

Type bool, optional

Inheritance



discordaio.enums

Classes

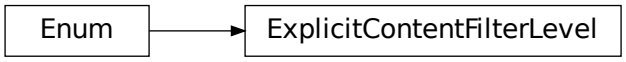
- *MessageNotificationLevel*: An enumeration.
- *ExplicitContentFilterLevel*: An enumeration.
- *MFALevel*: An enumeration.
- *VerificationLevel*: An enumeration.
- *ChannelTypes*: An enumeration.
- *MessageActivityTypes*: An enumeration.
- *GatewayOpcodes*: An enumeration.

class discordaio.enums.**MessageNotificationLevel**
An enumeration.

Inheritance

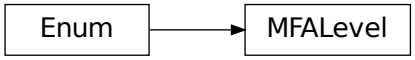
class discordaio.enums.**ExplicitContentFilterLevel**
An enumeration.

Inheritance



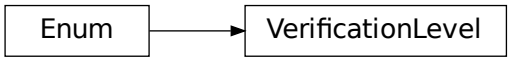
class discordaio.enums.**MFALevel**
An enumeration.

Inheritance



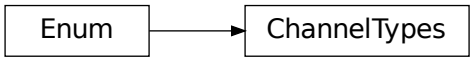
class discordaio.enums.**VerificationLevel**
An enumeration.

Inheritance



class discordaio.enums.**ChannelTypes**
An enumeration.

Inheritance



class discordaio.enums.**MessageActivityTypes**
An enumeration.

Inheritance

class discordaio.enums.**GatewayOpcodes**
An enumeration.

Inheritance

discordaio.exceptions

Exceptions

- *WebSocketCreationError*: Common base class for all non-exit exceptions.
- *EventTypeError*: Common base class for all non-exit exceptions.

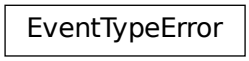
- *AuthorizationError*: Common base class for all non-exit exceptions.
- *UnhandledEndpointStatusError*: Common base class for all non-exit exceptions.

exception discordaio.exceptions.**WebSocketCreationError**

Inheritance

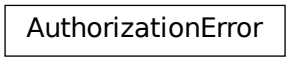
exception discordaio.exceptions.**EventTypeError**

Inheritance



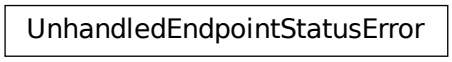
exception discordaio.exceptions.**AuthorizationError**(*message*)

Inheritance



exception discordaio.exceptions.**UnhandledEndpointStatusError**

Inheritance



discordaio.guild

Classes

- *Guild*: Represents a guild
- *GuildMember*: Represents a guild member
- *GuildEmbed*: Represents a guild embed
- *IntegrationAccount*: Represents a integration account
- *Integration*: Represents a integration
- *Ban*: Represents a ban

class discordaio.guild.**Guild**(*id=0*, *name=''*, *icon=''*, *splash=''*, *owner=False*, *owner_id=0*, *permissions=0*, *region=''*, *afk_channel_id=0*, *afk_timeout=0*, *embed_enabled=False*, *embed_channel_id=0*, *verification_level=0*, *default_message_notifications=0*, *explicit_content_filter=0*, *roles=[]*, *emojis=[]*, *features=[]*, *mfa_level=0*, *application_id=0*, *widget_enabled=False*, *widget_channel_id=0*, *system_channel_id=0*, *joined_at=None*, *large=None*, *unavailable=None*, *member_count=None*, *voice_states=None*, *members=[]*, *channels=None*, *presences=None*)

Represents a guild

New in version 0.2.0.

Note: Guilds in Discord represent an isolated collection of users and channels, and are often referred to as "servers" in the UI.

---

id
guild id

Type int

name
guild name (2-100 characters)

Type str

icon
icon hash

Type str

splash
splash hash

Type str

owner
whether or not the user is the owner of the guild

Type bool, optional

owner_id
id of owner

Type int

permissions
total permissions for the user in the guild (does not include channel overrides)

Type int, optional

region
voice region id for the guild

Type str

afk_channel_id
id of afk channel

Type int

afk_timeout
afk timeout in seconds

Type int

embed_enabled
is this guild embeddable (e.g. widget)

Type bool, optional

embed_channel_id
id of embedded channel

Type int, optional

verification_level
verification level required for the guild

Type int

default_message_notifications
default message notifications level

Type int

explicit_content_filter
explicit content filter level

Type int

roles
roles in the guild

Type list of *Role*

emojis
custom guild emojis

Type list of *Emoji*

features
enabled guild features

Type list of Strings

mfa_level
required MFA level for the guild

Type int

application_id
application id of the guild creator if it is bot-created

Type int

widget_enabled
whether or not the server widget is enabled

Type bool, optional

widget_channel_id
the channel id for the server widget

Type int, optional

system_channel_id
the id of the channel to which system messages are sent

Type int

joined_at
timestamp when this guild was joined at

Type int, optional

large
whether this is considered a large guild

Type bool, optional

unavailable
is this guild unavailable

Type bool, optional

member_count
total number of members in this guild

Type int, optional

voice_states
(without the guild_id key)

Type list of Partial

members
users in the guild

Type list of *Guild*

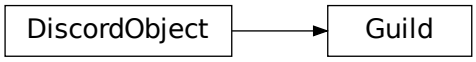
channels
channels in the guild

Type list of *Channel*

presences
presences of the users in the guild

Type list of Partial

Inheritance



await create_channel(*channel: discordaio.channel.Channel*) → discordaio.channel.Channel
Creates a new guild channel

New in version 0.3.0.

Parameters **channel** – The channel to create.

await delete() → None
Deletes the guild

New in version 0.3.0.

Raises AuthorizationError – Raised if you have no authorization to delete the guild.

await get_channels() → discordaio.channel.Channel
Returns a list of channels withing the guild.

New in version 0.3.0.

get_icon() → str
Returns the guild icon

New in version 0.2.0.

Returns The icon link

Return type str

await get_member(*member_id: int*) → discordaio.guild.GuildMember
Gets a guild member.

New in version 0.3.0.

Parameters **member_id** – The member id

await get_members()
Gets and fills the guild with the members info.

New in version 0.3.0.

get_splash()
Returns the guild splash

New in version 0.2.0.

Returns The splash link

Return type str

is_owner(*member: discordaio.guild.GuildMember*) → bool
Returns wether the guild member is the owner of the guild

New in version 0.2.0.

Parameters **member** (*GuildMember*) – The member

Returns True if it's the owner, False otherwise.

Return type bool

await leave()
Leaves a guild.

New in version 0.3.0.

class discordaio.guild.**GuildMember**(*user=<User Object: #, 0>, nick='', roles=[], joined_at=None, deaf=False, mute=False*)
Represents a guild member

New in version 0.2.0.

user
user object

Type *User*

nick
this users guild nickname (if one is set)

Type str, optional

roles
array of role object ids

Type list of int

joined_at
timestamp when the user joined the guild

Type int

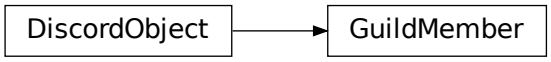
deaf
if the user is deafened

Type bool

mute
if the user is muted

Type bool

Inheritance



class discordaio.guild.**GuildEmbed**(*enabled=False*, *channel_id=0*)

Represents a guild embed

New in version 0.2.0.

enabled

if the embed is enabled

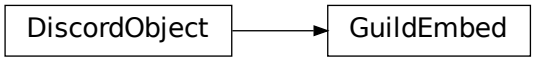
Type bool

channel_id

the embed channel id

Type int

Inheritance



class discordaio.guild.**IntegrationAccount**(*id=''*, *name=''*)

Represents a integration account

New in version 0.2.0.

id

id of the account

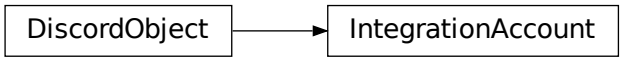
Type str

name

name of the account

Type str

Inheritance



class discordaio.guild.**Integration**(*id=0*, *name=''*, *type=''*, *enabled=False*, *syncing=False*, *role_id=0*, *expire_behavior=0*, *expire_grace_period=0*, *user=None*, *account=None*, *synced_at=None*)

Represents a integration

New in version 0.2.0.

id
integration id

Type int

name
integration name

Type str

type
integration type (twitch, youtube, etc)

Type str

enabled
is this integration enabled

Type bool

syncing
is this integration syncing

Type bool

role_id
id that this integration uses for "subscribers"

Type int

expire_behavior
the behavior of expiring subscribers

Type int

expire_grace_period
the grace period before expiring subscribers

Type int

user
object user for this integration

Type *User*

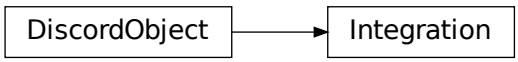
account
account information

Type Account

synced_at
timestamp when this integration was last synced

Type int

Inheritance



class discordaio.guild.**Ban**(*reason=''*, *user=None*)
Represents a ban

New in version 0.2.0.

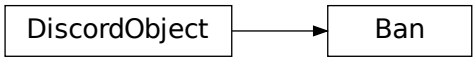
reason
the reason for the ban

Type str

user
the banned user

Type *User*

Inheritance



discordaio.http

- *Classes*

Classes

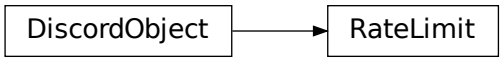
- *HTTPHandler*: Undocumented.
- *RateLimit*: Base class for discord objects.

class discordaio.http.**HTTPHandler**(*token*, *discord_client*)

Inheritance

class discordaio.http.**RateLimit**(*message=''*, *retry_after=0*, *_global=False*)

Inheritance



discordaio.invite

- *Classes*

Classes

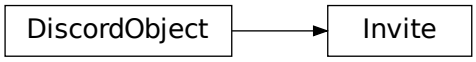
- *Invite*: Represents a code that when used, adds a user to a guild.

class discordaio.invite.**Invite**(*code=''*, *guild: Optional[discordaio.guild.Guild] = None*, *channel: Optional[discordaio.channel.Channel] = None*, *inviter: Optional[discordaio.user.User] = None*, *uses=0*, *max_uses=0*, *max_age=0*, *temporary=False*, *created_at=None*, *revoked=False*)

Represents a code that when used, adds a user to a guild.

New in version 0.2.0.

Inheritance



discordaio.permissions

TODO: do this, remember to do: channel edit permissions endpoint https://discordapp.com/developers/docs/topics/permissions#permissions

discordaio.role

Classes

- *Role*: Represents a discord role

class discordaio.role.**Role**(*id=0*, *name=''*, *color=0*, *hoist=False*, *position=0*, *permissions=0*, *managed=False*, *mentionable=False*)

Represents a discord role

New in version 0.2.0.

id
role id

Type int

name
role name

Type str

color
integer representation of hexadecimal color code

Type int

hoist
if this role is pinned in the user listing

Type bool

position
position of this role

Type int

permissions
permission bit set

Type int

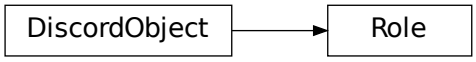
managed
whether this role is managed by an integration

Type bool

mentionable
whether this role is mentionable

Type bool

Inheritance



discordaio.user

Users in Discord are generally considered the base entity. Users can spawn across the entire platform, be members of guilds, participate in text and voice chat, and much more. Users are separated by a distinction of "bot" vs "normal." Although they are similar, bot users are automated users that are "owned" by another user. Unlike normal users, bot users do not have a limitation on the number of Guilds they can be a part of.

Classes

- *User*: Represents a discord user
- *UserConnection*: Represents a discord user connection

class discordaio.user.**User**(*id=0*, *username=''*, *discriminator=''*, *avatar=''*, *bot=False*, *system=False*, *mfa_enabled=False*, *locale=''*, *verified=False*, *email=''*, *flags=0*, *premium_type=0*, *public_flags=0*)

Represents a discord user

New in version 0.2.0.

id
the user's id identify

Type int

username
the user's username, not unique across the platform identify

Type str

discriminator
the user's 4-digit discord-tag identify

Type str

avatar
the user's avatar hash identify

Type str

bot
whether the user belongs to an OAuth2 application identify

Type bool, optional

system
whether the user is an Official Discord System user (part of the urgent message system)

Type bool, optional

mfa_enabled
whether the user has two factor enabled on their account identify

Type bool, optional

verified
whether the email on this account has been verified email

Type bool, optional

email
the user's email email

Type str, optional

flags
the flags on a user's account

Type int, optional

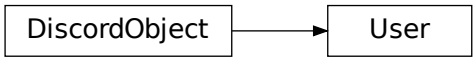
premium_type
the type of Nitro subscription on a user's account

Type int, optional

public_flags
the public flags on a user's account

Type int, optional

Inheritance



class discordaio.user.**UserConnection**(*id=''*, *name=''*, *type=''*, *revoked=False*, *integrations=[]*)

Represents a discord user connection

New in version 0.2.0.

id
id of the connection account

Type str

name
the username of the connection account

Type str

type
the service of the connection (twitch, youtube)

Type str

revoked
whether the connection is revoked

Type bool

integrations
an array of partial server integrations

Type list

Inheritance

discordaio.version

- *Variables*

Variables

- *__version__*

discordaio.version.**__version__**
str(object='') -> str str(bytes_or_buffer[, encoding[, errors]]) -> str

Create a new string object from the given object. If encoding or errors is specified, then the object must expose a data buffer that will be decoded using the given encoding and error handler. Otherwise, returns the result of object.__str__() (if defined) or repr(object). encoding defaults to sys.getdefaultencoding(). errors defaults to 'strict'. .. code-block:: text

'0.3.1'

discordaio.voice

- *Classes*

Classes

- *VoiceState*: Used to represent a user's voice connection status.
- *VoiceRegion*: Attributes:

class discordaio.voice.**VoiceState**(*guild_id: Optional[int] = None*, *channel_id: int = 0*, *user_id: int = 0*, *session_id: str = ''*, *deaf=False*, *mute=False*, *self_deaf=False*, *self_mute=False*, *suppress=False*)

Used to represent a user's voice connection status.

guild_id
the guild id this voice state is for

Type int, optional

channel_id
the channel id this user is connected to

Type int

user_id
the user id this voice state is for

Type int

session_id
the session id for this voice state

Type str

deaf
whether this user is deafened by the server

Type bool

mute
whether this user is muted by the server

Type bool

self_deaf
whether this user is locally deafened

Type bool

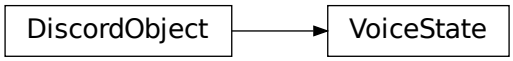
self_mute
whether this user is locally muted

Type bool

suppress
whether this user is muted by the current user

Type bool

Inheritance



class discordaio.voice.**VoiceRegion**(*id=''*, *name=''*, *vip=False*, *optimal=False*, *deprecated=False*, *custom=False*)

id
unique ID for the region

Type str

name
name of the region

Type str

vip
true if this is a vip-only server

Type bool

optimal
true for a single server that is closest to the current user's client

Type bool

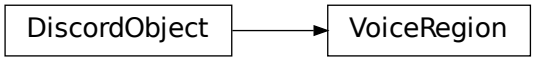
deprecated
whether this is a deprecated voice region (avoid switching to these)

Type bool

custom
whether this is a custom voice region (used for events/etc)

Type bool

Inheritance



discordaio.webhook

Webhooks are a low-effort way to post messages to channels in Discord. They do not require a bot user or authentication to use.

Classes

- *Webhook*: Used to represent a webhook.

class discordaio.webhook.**Webhook**(*id=0*, *guild_id=0*, *channel_id=0*, *user=None*, *name=''*, *avatar=''*, *token=''*)

Used to represent a webhook.

New in version 0.2.0.

id
the id of the webhook

Type int

guild_id
the guild id this webhook is for

Type int, optional

channel_id
the channel id this webhook is for

Type int

user
the user this webhook was created by (not returned when getting a webhook with its token)

Type *User*

name
the default name of the webhook

Type str

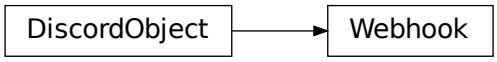
avatar
the default avatar of the webhook

Type str

token
the secure token of the webhook

Type str

Inheritance



discordaio.websocket

- *Classes*

Classes

- *DiscordWebsocket*: Class used for handling the websocket connection with the discord gateway

class discordaio.websocket.**DiscordWebsocket**(*http: discordaio.http.HTTPHandler = None*, *session_id: str = None*, *shards: list = []*)
Class used for handling the websocket connection with the discord gateway

New in version 0.2.0.

heartbeat_interval
The interval to send pings

Type float

_trace
Used for debugging

Type str

seq
Used in pings

Type str

session_id
Used for resuming

Type str

http
Used for sending http requests and session handling.

Type HTTPHandler

gateway_url
The gateway url

Type str

shards
Used for opening multiple connections

Type list of int

ws (
class:aiohttp.ClientWebSocketResponse‘‘): The websocket

Inheritance

await close() → bool
Closes the websocket

New in version 0.2.0.

Returns True if succeeded closing. False if the websocket was already closed

Return type bool

await start()
Starts the websocket

New in version 0.2.0.

1.5.4 Changelog

Version 0.4.0 (dev)

- Updated to discord api v8

Version 0.3.1

- Fixed async error when closing bot using ctrl-c
- Added Channel#send_message
- Added Channel#mention

- Added Channel#typing
- Added Channel#delete
- Added Channel#update
- Added Channel#refresh
- Added Channel#get_message
- Added ChannelMessage#delete_own_reaction
- Added ChannelMessage#delete_user_reaction
- Added ChannelMessage#delete_all_reactions
- Added ChannelMessage#get_reactions
- Added ChannelMessage#update
- Added ChannelMessage#delete
- Added ChannelMessage#bulk_delete_messages
- Added ChannelMessage#get_invites
- Added ChannelMessage#create_invite
- Added ChannelMessage#delete_permission
- Added ChannelMessage#get_pinned_messages
- Added ChannelMessage#pin_message
- Added ChannelMessage#delete_pinned_message
- Moved DiscordBot#get_messages to Channel#get_messages
- Moved DiscordBot#leave_guild to Guild#leave
- Moved DiscordBot#get_guild_member to Guild#get_member
- Moved DiscordBot#get_guild_members to Guild#get_members
- Moved DiscordBot#create_guild_channel to Guild#create_channel
- Moved DiscordBot#delete_guild to Guild#delete
- Moved DiscordBot#get_guild_channels to Guild#get_channels
- Moved DiscordBot#modify_channel to Channel#update

Version 0.2.2

- Added DiscordBot.get_guild_channels method
- Added DiscordBot.create_guild_channel method
- Added DiscordBot.delete_guild method
- Added DiscordBot.move_channels method
- Added NotFoundError exception
- Added GatewayUnavailable exception
- Added BadRequestError exception
- Added AuthorizationError exception

- Fixed a bug in websocket

Version 0.2.0

- All discord events can be used now.

Version 0.1.0

- Initial development version

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

d

Index

Symbols

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W