discord.aio Documentation

Release 0.2.1

Ryozuki

Feb 25, 2018

Contents:

1 discord.aio **3**
1.1 Documentation 3
1.2 Installation 3
1.3 Local development 3
1.4 Example bot 4
1.5 TODO 4

2 Indices and tables **51**

Python Module Index **53**

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}')
```

Here you can find a more extensive example.

1.5 TODO

- Add compression support
- Add bot shards support
- Handle ISO8601 timestamp

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*

Submodules

discordaio.activity

- *Classes*

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
: `str` – the activity’s name

type
: `int` – activity type

url
: `str`, optional – stream url, is validated when type is 1

timestamps
: `Timestamps` – object unix timestamps for start and/or end of the game

application_id
: `int`, optional – application id for the game

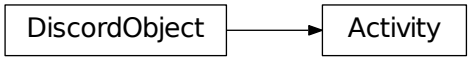
details
: `str`, optional – what the player is currently doing

state
: `str`, optional – the user’s current party status

party
: `Party` – object information for the current party of the player

assets
: `Assets` – object images for the presence and their hover texts

Inheritance



class discordaio.activity.**ActivityParty**(*id=”, size=[]*)

Activity party

New in version 0.2.0.

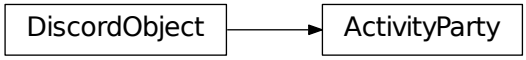
id

str, optional – the id of the party

size

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

Inheritance



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

Activity timestamps

New in version 0.2.0.

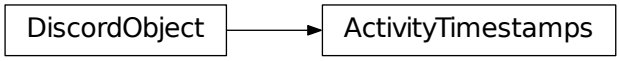
start

int, optional – unix time (in milliseconds) of when the activity started

end

int, optional – unix time (in milliseconds) of when the activity ends

Inheritance



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

Activity assets

New in version 0.2.0.

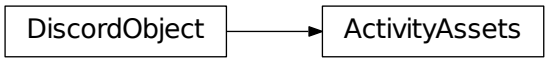
large_image
: str, optional – the id for a large asset of the activity, usually a snowflake

large_text
: str, optional – text displayed when hovering over the large image of the activity

small_image
: str, optional – the id for a small asset of the activity, usually a snowflake

small_text
: str, optional – text displayed when hovering over the small image of the activity

Inheritance



discordaio.base

- *Classes*

Classes

- *DiscordObject*: Base class for discord objects.

class discordaio.base.**DiscordObject**

Base class for discord objects.

Inheritance

DiscordObject

classmethod await from_api_res(*coro_or_json_or_str*)
Parses a discord API response

discordaio.channel

Contains all related Channel Discord objects

- *Classes*

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=0, type=0, guild_id=0, position=0, permission_overwrites=[], name="", topic="", nsfw=False, last_message_id=0, bitrate=0, user_limit=0, recipients=[], icon="", owner_id=0, application_id=0, parent_id=0, last_pin_timestamp=None*)

Represents a guild or DM channel within Discord.

New in version 0.2.0.

id
int – the id of this channel

value_type
int – the value_type of channel

guild_id
int, optional – the id of the guild

position
int, optional – sorting position of the channel

permission_overwrites
list of *Overwrite*, optional – explicit permission overwrites for members and roles

name
str, optional – the name of the channel (2-100 characters)

topic
str, optional – the channel topic (0-1024 characters)

nsfw
bool, optional – if the channel is nsfw

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

bitrate
int, optional – the bitrate (in bits) of the voice channel

user_limit
int, optional – the user limit of the voice channel

recipients
list of *User*, optional – the recipients of the DM

icon
str, optional – icon hash

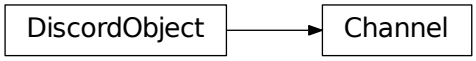
owner_id
int, optional – id of the DM creator

application_id
int, optional – application id of the group DM creator if it is bot-created

parent_id
int, optional – id of the parent category for a channel

last_pin_timestamp
int, optional – timestamp when the last pinned message was pinned

Inheritance



class discordaio.channel.**ChannelMessage**(*id=0, channel_id=0, author=None, content=", timestamp=None, edited_timestamp=None, tts=False, mention_everyone=False, mentions=[], mention_roles=[], attachments=[], embeds=[], reactions=[], nonce=0, pinned=False, webhook_id=0, type=0, 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
`int` – id of the message

channel_id
`int` – id of the channel the message was sent in

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

content
`str` – contents of the message

timestamp
`int` – timestamp when this message was sent

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

tts
`bool` – whether this was a TTS message

mention_everyone
`bool` – whether this message mentions everyone

mentions
`list` of `User` – objects users specifically mentioned in the message

mention_roles
`list` of *`Role`* – object ids roles specifically mentioned in this message

attachments
`list` of *`Attachment`* – objects any attached files

embeds
`list` of *`Embed`* – objects any embedded content

reactions
`list` of *`Reaction`* – objects reactions to the message

nonce
`int`, optional – used for validating a message was sent

pinned
`bool` – whether this message is pinned

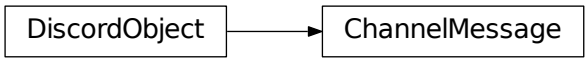
webhook_id
`int`, optional – if the message is generated by a webhook, this is the webhook's id

type
`int` – type of message

activity
MessageActivity – activity object sent with Rich Presence-related chat embeds

application
MessageApplication – application object sent with Rich Presence-related chat embeds

Inheritance



class `discordaio.channel.`**`Overwrite`**(*id=0*, *type=''*, *allow=0*, *deny=0*)

Represents a Overwrite object.

New in version 0.2.0.

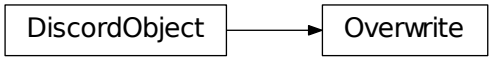
id
`int` – role or user id

type
`str` – either "role" or "member"

allow
`int` – permission bit set

deny
`int` – permission bit set

Inheritance



class discordaio.channel.**MessageActivity**(*type=None*, *party_id=''*)

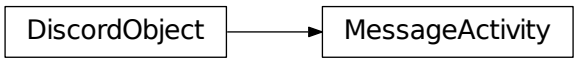
Represents a Message Activity.

New in version 0.2.0.

type
: `int` – type of message activity

party_id
: `str`, optional – party_id from a Rich Presence event

Inheritance



class discordaio.channel.**MessageApplication**(*id=0*, *cover_image=''*, *description=''*, *icon=''*, *name=''*)

Represents a Message Application.

New in version 0.2.0.

id
: `int` – id of the application

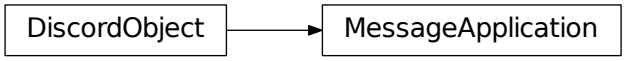
cover_image
: `str` – id of the embed's image asset

description
: `str` – application's description

icon
: `str` – id of the application's icon

name
: `str` – name of the application

Inheritance



class discordaio.channel.**Reaction**(*count=0*, *me=False*, *emoji=None*)

Represents a Reaction.

New in version 0.2.0.

count

int – times this emoji has been used to react

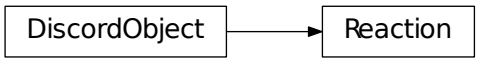
me

bool – whether the current user reacted using this emoji

emoji

Emoji – emoji information

Inheritance



class discordaio.channel.**Embed**(*title=''*, *type=''*, *description=''*, *url=''*, *timestamp=None*, *color=0*, *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

str – title of embed

type

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

description

str – description of embed

url
str – url of embed

timestamp
int – timestamp of embed content

color
int – color code of the embed

footer
EmbedFooter – footer information

image
EmbedImage – image information

thumbnail
EmbedThumbnail – thumbnail information

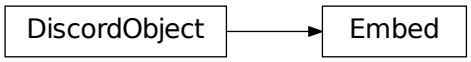
video
EmbedVideo – video information

provider
EmbedProvider – provider information

author
EmbedAuthor – author information

fields
list of *EmbedField* – fields information

Inheritance



class discordaio.channel.**EmbedThumbnail**(*url=”, proxy_url=”, height=0, width=0*)

Represents a embed thumbnail object

New in version 0.2.0.

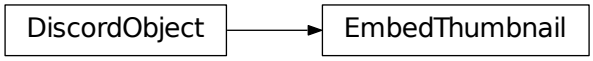
url
str – source url of thumbnail (only supports http(s) and attachments)

proxy_url
str – a proxied url of the thumbnail

height
int – height of thumbnail

width
int – width of thumbnail

Inheritance



class `discordaio.channel.`**`EmbedVideo`**(*url=”, height=0, width=0*)

Represents a embed video

New in version 0.2.0.

`url`

`str` – source url of video

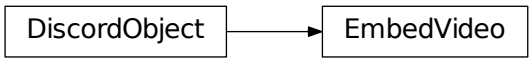
`height`

`int` – height of video

`width`

`int` – width of video

Inheritance



class `discordaio.channel.`**`EmbedImage`**(*url=”, proxy_url=”, height=0, width=0*)

Represents a embed image

New in version 0.2.0.

`url`

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

`proxy_url`

`str` – a proxied url of the image

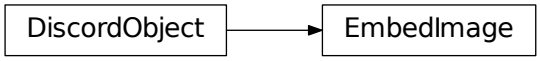
`height`

`int` – height of image

`width`

`int` – width of image

Inheritance



class discordaio.channel.**EmbedProvider**(*name=”*, *url=”*)

Represents a embed provider

New in version 0.2.0.

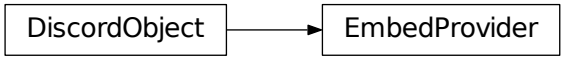
name

str – name of provider

url

str – url of provider

Inheritance



class discordaio.channel.**EmbedAuthor**(*name=”*, *url=”*, *icon_url=”*, *proxy_icon_url=”*)

Represents a embed author

New in version 0.2.0.

name

str – name of author

url

str – url of author

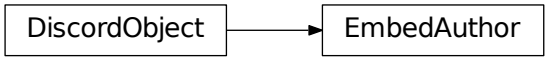
icon_url

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

proxy_icon_url

str – a proxied url of author icon

Inheritance



class discordaio.channel.**EmbedFooter**(*text=”*, *icon_url=”*, *proxy_icon_url=”*)

Represents a embed footer

New in version 0.2.0.

text

`str` – footer text

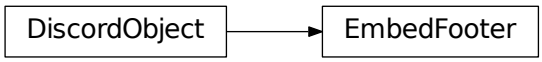
icon_url

`str` – url of footer icon (only supports http(s) and attachments)

proxy_icon_url

`str` – a proxied url of footer icon

Inheritance



class discordaio.channel.**EmbedField**(*name=”*, *value=”*, *inline=False*)

Represents a embed field

New in version 0.2.0.

name

`str` – name of the field

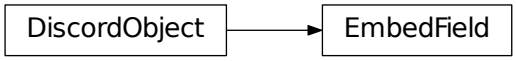
value

`str` – value of the field

inline

`bool` – whether or not this field should display inline

Inheritance



class discordaio.channel.**Attachment**(*id=0*, *filename=''*, *size=0*, *url=''*, *proxy_url=''*, *height=0*, *width=0*)

Represents a attachment

New in version 0.2.0.

id
: int – attachment id

filename
: str – name of file attached

size
: int – size of file in bytes

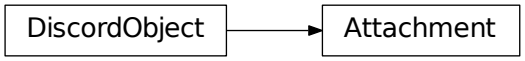
url
: str – source url of file

proxy_url
: str – a proxied url of file

height
: int – height of file (if image)

width
: int – width of file (if image)

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

str – The discord token used for authentication

http

HTTPHandler – Used for making http requests and websocket creation.

guilds

list of *Guild* – The list of guilds the bot is in.

user

User – The user object of the bot.

ws

DiscordWebsocket – The websocket used for communication

Inheritance

await delete_channel(*channel_id: int*) → discordaio.channel.Channel

Deletes a channel.

Note: Delete a channel, or close a private message. Requires the 'MANAGE_CHANNELS' permission for the guild. Deleting a category does not delete its child channels; they will have their parent_id removed and a Channel Update Gateway event will fire for each of them. Returns a channel object on success. Fires a Channel Delete Gateway event.

New in version 0.2.0.

Parameters **channel_id** (int) – The channel id

Returns The deleted channel

Return type *Channel*

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** (int) – The channel id

Returns The channel

Return type *Channel*

await get_dms() → list
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_guild_member(*guild: discordaio.guild.Guild*, *member_id: int*) → discordaio.guild.GuildMember
Gets a guild member info for the guild and the member id.

New in version 0.2.0.

Parameters

- **guild** (*Guild*) – The guild
- **member_id** (int) – The member id

Returns The guild member

Return type *GuildMember*

await get_guild_members(*guild: discordaio.guild.Guild*)
Gets and fills the guild with the members info.

New in version 0.2.0.

Parameters **guild** (*Guild*) – The guild to fill the members.

await get_guilds() → list
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: int*) → 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 leave_guild(*guild_id: int*)
Leaves a guild.

New in version 0.2.0.

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

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:: guess

'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:: guess

'https://discordapp.com/api/v6'

discordaio.emoji

- *Classes*

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
int – emoji id

name
str – emoji name

roles
list of *Role* – object ids roles this emoji is whitelisted to

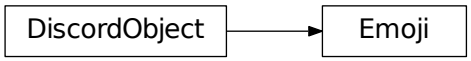
user
User – object user that created this emoji

require_colons
bool, optional – whether this emoji must be wrapped in colons

managed
bool, optional – whether this emoji is managed

animated
bool, optional – whether this emoji is animated

Inheritance



discordaio.enums

- *Classes*

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**

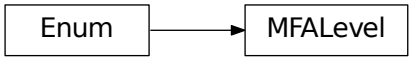
An enumeration.

Inheritance

class discordaio.enums.**MFALevel**

An enumeration.

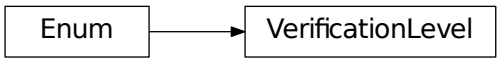
Inheritance



class discordaio.enums.**VerificationLevel**

An enumeration.

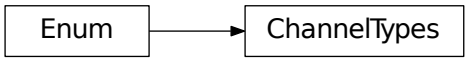
Inheritance



class discordaio.enums.**ChannelTypes**

An enumeration.

Inheritance



class discordaio.enums.**MessageActivityTypes**

An enumeration.

Inheritance

class discordaio.enums.**GatewayOpcodes**

An enumeration.

Inheritance

discordaio.exceptions

- *Exceptions*

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.

exception discordaio.exceptions.**WebSocketCreationError**

Inheritance

WebSocketCreationError

exception discordaio.exceptions.**EventTypeError**

Inheritance

EventTypeError

exception discordaio.exceptions.**AuthorizationError**

Inheritance

AuthorizationError

discordaio.guild

- *Classes*

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
int – guild id

name
str – guild name (2-100 characters)

icon
str – icon hash

splash
str – splash hash

owner
bool, optional – whether or not the user is the owner of the guild

owner_id
int – id of owner

permissions
int, optional – total permissions for the user in the guild (does not include channel overrides)

region
str – voice region id for the guild

afk_channel_id
int – id of afk channel

afk_timeout
int – afk timeout in seconds

embed_enabled
bool, optional – is this guild embeddable (e.g. widget)

embed_channel_id
int, optional – id of embedded channel

verification_level
int – verification level required for the guild

default_message_notifications
`int` – default message notifications level

explicit_content_filter
`int` – explicit content filter level

roles
`list` of *`Role`* – roles in the guild

emojis
`list` of *`Emoji`* – custom guild emojis

features
`list` of `Strings` – enabled guild features

mfa_level
`int` – required MFA level for the guild

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

widget_enabled
`bool`, optional – whether or not the server widget is enabled

widget_channel_id
`int`, optional – the channel id for the server widget

system_channel_id
`int` – the id of the channel to which system messages are sent

joined_at
`int`, optional – timestamp when this guild was joined at

large
`bool`, optional – whether this is considered a large guild

unavailable
`bool`, optional – is this guild unavailable

member_count
`int`, optional – total number of members in this guild

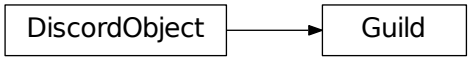
voice_states
`list` of `Partial` – (without the guild_id key)

members
`list` of *`Guild`* – users in the guild

channels
`list` of *`Channel`* – channels in the guild

presences
`list` of `Partial` – presences of the users in the guild

Inheritance



get_icon() → str

Returns the guild icon

New in version 0.2.0.

Returns The icon link

Return type str

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

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 – user object

nick

str, optional – this users guild nickname (if one is set)

roles

list of int – array of role object ids

joined_at

int – timestamp when the user joined the guild

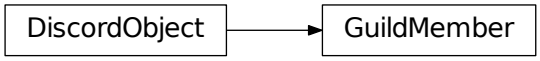
deaf

bool – if the user is deafened

mute

bool – if the user is muted

Inheritance



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

Represents a guild embed

New in version 0.2.0.

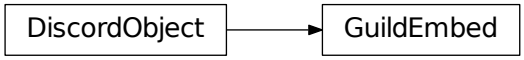
enabled

bool – if the embed is enabled

channel_id

int – the embed channel id

Inheritance



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

Represents a integration account

New in version 0.2.0.

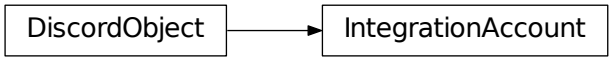
id

str – id of the account

name

str – name of the account

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
: `int` – integration id

name
: `str` – integration name

type
: `str` – integration type (twitch, youtube, etc)

enabled
: `bool` – is this integration enabled

syncing
: `bool` – is this integration syncing

role_id
: `int` – id that this integration uses for "subscribers"

expire_behavior
: `int` – the behavior of expiring subscribers

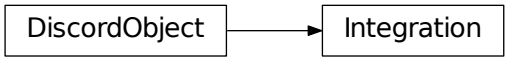
expire_grace_period
: `int` – the grace period before expiring subscribers

user
: *`User`* – object user for this integration

account
: `Account` – account information

synced_at
: `int` – timestamp when this integration was last synced

Inheritance



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

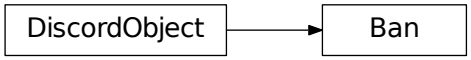
Represents a ban

New in version 0.2.0.

reason
: `str` – the reason for the ban

user
User – the banned user

Inheritance



discordaio.http

- *Classes*

Classes

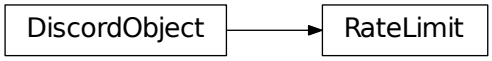
- *HTTPHandler*: Undocumented.
- *RateLimit*: Base class for discord objects.

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.
- `InviteMetadata`: Represents the invite metadata

class `discordaio.invite.`**`Invite`**(*code=''*, *guild=None*, *channel=None*)

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

New in version 0.2.0.

code

`str` – the invite code (unique ID)

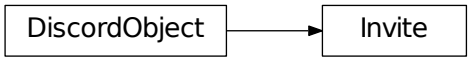
guild

`Guild` – the guild this invite is for

channel

`Channel` – the channel this invite is for

Inheritance



class `discordaio.invite.`**`InviteMetadata`**(*inviter=None*, *uses=0*, *max_uses=0*, *max_age=0*, *temporary=False*, *created_at=None*, *revoked=False*)

Represents the invite metadata

New in version 0.2.0.

inviter
A – user object user who created the invite

uses
int – number of times this invite has been used

max_uses
int – max number of times this invite can be used

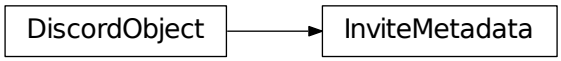
max_age
int – duration (in seconds) after which the invite expires

temporary
bool – whether this invite only grants temporary membership

created_at
int – timestamp when this invite was created

revoked
bool – whether this invite is revoked

Inheritance



discordaio.role

- *Classes*

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
int – role id

name
str – role name

color
`int` – integer representation of hexadecimal color code

hoist
`bool` – if this role is pinned in the user listing

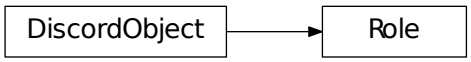
position
`int` – position of this role

permissions
`int` – permission bit set

managed
`bool` – whether this role is managed by an integration

mentionable
`bool` – whether this role is mentionable

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*

Classes

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

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

Represents a discord user

New in version 0.2.0.

id
`int` – the user's id identify

username
`str` – the user's username, not unique across the platform identify

discriminator
`str` – the user's 4-digit discord-tag identify

avatar
`str` – the user's avatar hash identify

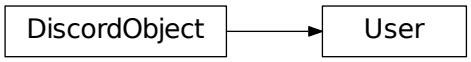
bot
`bool`, optional – whether the user belongs to an OAuth2 application identify

mfa_enabled
`bool`, optional – whether the user has two factor enabled on their account identify

verified
`bool`, optional – whether the email on this account has been verified email

email
`str`, optional – the user's email email

Inheritance



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

Represents a discord user connection

New in version 0.2.0.

id
`str` – id of the connection account

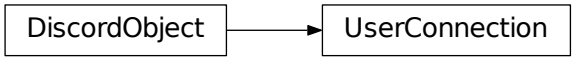
name
`str` – the username of the connection account

type
`str` – the service of the connection (twitch, youtube)

revoked
`bool` – whether the connection is revoked

integrations
`list` – an array of partial server integrations

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:: guess

'0.2.1'

discordaio.voice

- *Classes*

Classes

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

class `discordaio.voice.VoiceState`(*guild_id: typing.Union[int, NoneType] = 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

`int`, optional – the guild id this voice state is for

channel_id
`int` – the channel id this user is connected to

user_id
`int` – the user id this voice state is for

session_id
`str` – the session id for this voice state

deaf
`bool` – whether this user is deafened by the server

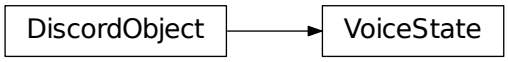
mute
`bool` – whether this user is muted by the server

self_deaf
`bool` – whether this user is locally deafened

self_mute
`bool` – whether this user is locally muted

suppress
`bool` – whether this user is muted by the current user

Inheritance



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

id
`str` – unique ID for the region

name
`str` – name of the region

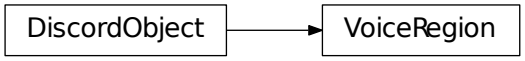
vip
`bool` – true if this is a vip-only server

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

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

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

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*

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
`int` – the id of the webhook

guild_id
`int`, optional – the guild id this webhook is for

channel_id
`int` – the channel id this webhook is for

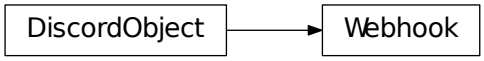
user
`User` – the user this webhook was created by (not returned when getting a webhook with its token)

name
`str` – the default name of the webhook

avatar
`str` – the default avatar of the webhook

token
`str` – the secure token of the webhook

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
`float` – The interval to send pings

_trace
`str` – Used for debugging

seq
`str` – Used in pings

session_id
`str` – Used for resuming

http
`HTTPHandler` – Used for sending http requests and session handling.

gateway_url
`str` – The gateway url

shards
`list` of `int` – Used for opening multiple connections

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

Inheritance

`await close`() → bool

Closes the websocket

New in version 0.2.0.

Returns True if succeded 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 0.2.0

- 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

discordaio, 12
discordaio.activity, 13
discordaio.base, 15
discordaio.channel, 16
discordaio.client, 26
discordaio.constants, 29
discordaio.emoji, 29
discordaio.enums, 30
discordaio.exceptions, 33
discordaio.guild, 34
discordaio.http, 40
discordaio.invite, 41
discordaio.role, 42
discordaio.user, 43
discordaio.version, 45
discordaio.voice, 45
discordaio.webhook, 47
discordaio.websocket, 48

Index

Symbols

__version__ (in module discordaio.version), 45
_trace (discordaio.websocket.DiscordWebsocket attribute), 48

A

account (discordaio.guild.Integration attribute), 39
Activity (class in discordaio.activity), 13
activity (discordaio.channel.ChannelMessage attribute), 19
ActivityAssets (class in discordaio.activity), 15
ActivityParty (class in discordaio.activity), 14
ActivityTimestamps (class in discordaio.activity), 14
afk_channel_id (discordaio.guild.Guild attribute), 35
afk_timeout (discordaio.guild.Guild attribute), 35
allow (discordaio.channel.Overwrite attribute), 19
animated (discordaio.emoji.Emoji attribute), 30
application (discordaio.channel.ChannelMessage attribute), 19
application_id (discordaio.activity.Activity attribute), 13
application_id (discordaio.channel.Channel attribute), 17
application_id (discordaio.guild.Guild attribute), 36
assets (discordaio.activity.Activity attribute), 13
Attachment (class in discordaio.channel), 26
attachments (discordaio.channel.ChannelMessage attribute), 19
author (discordaio.channel.ChannelMessage attribute), 18
author (discordaio.channel.Embed attribute), 22
AuthorizationError, 34
avatar (discordaio.user.User attribute), 44
avatar (discordaio.webhook.Webhook attribute), 47

B

Ban (class in discordaio.guild), 39
bitrate (discordaio.channel.Channel attribute), 17
bot (discordaio.user.User attribute), 44

C

Channel (class in discordaio.channel), 16
channel (discordaio.invite.Invite attribute), 41
channel_id (discordaio.channel.ChannelMessage attribute), 18
channel_id (discordaio.guild.GuildEmbed attribute), 38
channel_id (discordaio.voice.VoiceState attribute), 45
channel_id (discordaio.webhook.Webhook attribute), 47
ChannelMessage (class in discordaio.channel), 18
channels (discordaio.guild.Guild attribute), 36
ChannelTypes (class in discordaio.enums), 32
close() (discordaio.websocket.DiscordWebsocket method), 49
code (discordaio.invite.Invite attribute), 41
color (discordaio.channel.Embed attribute), 22
color (discordaio.role.Role attribute), 42
content (discordaio.channel.ChannelMessage attribute), 18
count (discordaio.channel.Reaction attribute), 21
cover_image (discordaio.channel.MessageApplication attribute), 20
created_at (discordaio.invite.InviteMetadata attribute), 42
custom (discordaio.voice.VoiceRegion attribute), 46

D

deaf (discordaio.guild.GuildMember attribute), 37
deaf (discordaio.voice.VoiceState attribute), 46
default_message_notifications (discordaio.guild.Guild attribute), 36
delete_channel() (discordaio.client.DiscordBot method), 27
deny (discordaio.channel.Overwrite attribute), 19
deprecated (discordaio.voice.VoiceRegion attribute), 46
description (discordaio.channel.Embed attribute), 21
description (discordaio.channel.MessageApplication attribute), 20
details (discordaio.activity.Activity attribute), 13
DISCORD_API_URL (in module discordaio.constants), 29
DISCORD_CDN (in module discordaio.constants), 29
discordaio (module), 12
discordaio.activity (module), 13

discordaio.base (module), 15
discordaio.channel (module), 16
discordaio.client (module), 26
discordaio.constants (module), 29
discordaio.emoji (module), 29
discordaio.enums (module), 30
discordaio.exceptions (module), 33
discordaio.guild (module), 34
discordaio.http (module), 40
discordaio.invite (module), 41
discordaio.role (module), 42
discordaio.user (module), 43
discordaio.version (module), 45
discordaio.voice (module), 45
discordaio.webhook (module), 47
discordaio.websocket (module), 48
DiscordBot (class in discordaio.client), 27
DiscordObject (class in discordaio.base), 15
DiscordWebsocket (class in discordaio.websocket), 48
discriminator (discordaio.user.User attribute), 44

E

edited_timestamp (discordaio.channel.ChannelMessage attribute), 18
email (discordaio.user.User attribute), 44
Embed (class in discordaio.channel), 21
embed_channel_id (discordaio.guild.Guild attribute), 35
embed_enabled (discordaio.guild.Guild attribute), 35
EmbedAuthor (class in discordaio.channel), 24
EmbedField (class in discordaio.channel), 25
EmbedFooter (class in discordaio.channel), 25
EmbedImage (class in discordaio.channel), 23
EmbedProvider (class in discordaio.channel), 24
embeds (discordaio.channel.ChannelMessage attribute), 19
EmbedThumbnail (class in discordaio.channel), 22
EmbedVideo (class in discordaio.channel), 23
Emoji (class in discordaio.emoji), 30
emoji (discordaio.channel.Reaction attribute), 21
emojis (discordaio.guild.Guild attribute), 36
enabled (discordaio.guild.GuildEmbed attribute), 38
enabled (discordaio.guild.Integration attribute), 39
end (discordaio.activity.ActivityTimestamps attribute), 14
event() (discordaio.client.DiscordBot method), 27
EventTypeError, 34
exit() (discordaio.client.DiscordBot method), 27
expire_behavior (discordaio.guild.Integration attribute), 39
expire_grace_period (discordaio.guild.Integration attribute), 39
explicit_content_filter (discordaio.guild.Guild attribute), 36
ExplicitContentFilterLevel (class in discordaio.enums), 31

F

features (discordaio.guild.Guild attribute), 36
fields (discordaio.channel.Embed attribute), 22
filename (discordaio.channel.Attachment attribute), 26
footer (discordaio.channel.Embed attribute), 22
from_api_res() (discordaio.base.DiscordObject method), 16

G

gateway_url (discordaio.websocket.DiscordWebsocket attribute), 48
GatewayOpcodes (class in discordaio.enums), 33
get_channel() (discordaio.client.DiscordBot method), 28
get_dms() (discordaio.client.DiscordBot method), 28
get_guild() (discordaio.client.DiscordBot method), 28
get_guild_member() (discordaio.client.DiscordBot method), 28
get_guild_members() (discordaio.client.DiscordBot method), 28
get_guilds() (discordaio.client.DiscordBot method), 28
get_icon() (discordaio.guild.Guild method), 37
get_self_user() (discordaio.client.DiscordBot method), 28
get_splash() (discordaio.guild.Guild method), 37
get_user() (discordaio.client.DiscordBot method), 28
Guild (class in discordaio.guild), 35
guild (discordaio.invite.Invite attribute), 41
guild_id (discordaio.channel.Channel attribute), 17
guild_id (discordaio.voice.VoiceState attribute), 45
guild_id (discordaio.webhook.Webhook attribute), 47
GuildEmbed (class in discordaio.guild), 38
GuildMember (class in discordaio.guild), 37
guilds (discordaio.client.DiscordBot attribute), 27

H

heartbeat_interval (discordaio.websocket.DiscordWebsocket attribute), 48
height (discordaio.channel.Attachment attribute), 26
height (discordaio.channel.EmbedImage attribute), 23
height (discordaio.channel.EmbedThumbnail attribute), 22
height (discordaio.channel.EmbedVideo attribute), 23
hoist (discordaio.role.Role attribute), 43
http (discordaio.client.DiscordBot attribute), 27
http (discordaio.websocket.DiscordWebsocket attribute), 48
HTTPHandler (class in discordaio.http), 40

I

icon (discordaio.channel.Channel attribute), 17
icon (discordaio.channel.MessageApplication attribute), 20

icon (discordaio.guild.Guild attribute), 35
icon_url (discordaio.channel.EmbedAuthor attribute), 24
icon_url (discordaio.channel.EmbedFooter attribute), 25
id (discordaio.activity.ActivityParty attribute), 14
id (discordaio.channel.Attachment attribute), 26
id (discordaio.channel.Channel attribute), 17
id (discordaio.channel.ChannelMessage attribute), 18
id (discordaio.channel.MessageApplication attribute), 20
id (discordaio.channel.Overwrite attribute), 19
id (discordaio.emoji.Emoji attribute), 30
id (discordaio.guild.Guild attribute), 35
id (discordaio.guild.Integration attribute), 39
id (discordaio.guild.IntegrationAccount attribute), 38
id (discordaio.role.Role attribute), 42
id (discordaio.user.User attribute), 43
id (discordaio.user.UserConnection attribute), 44
id (discordaio.voice.VoiceRegion attribute), 46
id (discordaio.webhook.Webhook attribute), 47
image (discordaio.channel.Embed attribute), 22
inline (discordaio.channel.EmbedField attribute), 25
Integration (class in discordaio.guild), 38
IntegrationAccount (class in discordaio.guild), 38
integrations (discordaio.user.UserConnection attribute), 44
Invite (class in discordaio.invite), 41
InviteMetadata (class in discordaio.invite), 41
inviter (discordaio.invite.InviteMetadata attribute), 42
is_owner() (discordaio.guild.Guild method), 37

J

joined_at (discordaio.guild.Guild attribute), 36
joined_at (discordaio.guild.GuildMember attribute), 37

L

large (discordaio.guild.Guild attribute), 36
large_image (discordaio.activity.ActivityAssets attribute), 15
large_text (discordaio.activity.ActivityAssets attribute), 15
last_message_id (discordaio.channel.Channel attribute), 17
last_pin_timestamp (discordaio.channel.Channel attribute), 17
leave_guild() (discordaio.client.DiscordBot method), 29

M

managed (discordaio.emoji.Emoji attribute), 30
managed (discordaio.role.Role attribute), 43
max_age (discordaio.invite.InviteMetadata attribute), 42
max_uses (discordaio.invite.InviteMetadata attribute), 42
me (discordaio.channel.Reaction attribute), 21
member_count (discordaio.guild.Guild attribute), 36
members (discordaio.guild.Guild attribute), 36
mention_everyone (discordaio.channel.ChannelMessage attribute), 18
mention_roles (discordaio.channel.ChannelMessage attribute), 18
mentionable (discordaio.role.Role attribute), 43
mentions (discordaio.channel.ChannelMessage attribute), 18
MessageActivity (class in discordaio.channel), 20
MessageActivityTypes (class in discordaio.enums), 32
MessageApplication (class in discordaio.channel), 20
MessageNotificationLevel (class in discordaio.enums), 31
mfa_enabled (discordaio.user.User attribute), 44
mfa_level (discordaio.guild.Guild attribute), 36
MFALevel (class in discordaio.enums), 31
mute (discordaio.guild.GuildMember attribute), 37
mute (discordaio.voice.VoiceState attribute), 46

N

name (discordaio.activity.Activity attribute), 13
name (discordaio.channel.Channel attribute), 17
name (discordaio.channel.EmbedAuthor attribute), 24
name (discordaio.channel.EmbedField attribute), 25
name (discordaio.channel.EmbedProvider attribute), 24
name (discordaio.channel.MessageApplication attribute), 20
name (discordaio.emoji.Emoji attribute), 30
name (discordaio.guild.Guild attribute), 35
name (discordaio.guild.Integration attribute), 39
name (discordaio.guild.IntegrationAccount attribute), 38
name (discordaio.role.Role attribute), 42
name (discordaio.user.UserConnection attribute), 44
name (discordaio.voice.VoiceRegion attribute), 46
name (discordaio.webhook.Webhook attribute), 47
nick (discordaio.guild.GuildMember attribute), 37
nonce (discordaio.channel.ChannelMessage attribute), 19
nsfw (discordaio.channel.Channel attribute), 17

O

optimal (discordaio.voice.VoiceRegion attribute), 46
Overwrite (class in discordaio.channel), 19
owner (discordaio.guild.Guild attribute), 35
owner_id (discordaio.channel.Channel attribute), 17
owner_id (discordaio.guild.Guild attribute), 35

P

parent_id (discordaio.channel.Channel attribute), 17
party (discordaio.activity.Activity attribute), 13
party_id (discordaio.channel.MessageActivity attribute), 20
permission_overwrites (discordaio.channel.Channel attribute), 17
permissions (discordaio.guild.Guild attribute), 35
permissions (discordaio.role.Role attribute), 43

pinned (discordaio.channel.ChannelMessage attribute), 19
position (discordaio.channel.Channel attribute), 17
position (discordaio.role.Role attribute), 43
presences (discordaio.guild.Guild attribute), 36
provider (discordaio.channel.Embed attribute), 22
proxy_icon_url (discordaio.channel.EmbedAuthor attribute), 24
proxy_icon_url (discordaio.channel.EmbedFooter attribute), 25
proxy_url (discordaio.channel.Attachment attribute), 26
proxy_url (discordaio.channel.EmbedImage attribute), 23
proxy_url (discordaio.channel.EmbedThumbnail attribute), 22

R

RateLimit (class in discordaio.http), 40
Reaction (class in discordaio.channel), 21
reactions (discordaio.channel.ChannelMessage attribute), 19
reason (discordaio.guild.Ban attribute), 39
recipients (discordaio.channel.Channel attribute), 17
region (discordaio.guild.Guild attribute), 35
require_colons (discordaio.emoji.Emoji attribute), 30
revoked (discordaio.invite.InviteMetadata attribute), 42
revoked (discordaio.user.UserConnection attribute), 44
Role (class in discordaio.role), 42
role_id (discordaio.guild.Integration attribute), 39
roles (discordaio.emoji.Emoji attribute), 30
roles (discordaio.guild.Guild attribute), 36
roles (discordaio.guild.GuildMember attribute), 37
run() (discordaio.client.DiscordBot method), 29

S

self_deaf (discordaio.voice.VoiceState attribute), 46
self_mute (discordaio.voice.VoiceState attribute), 46
seq (discordaio.websocket.DiscordWebsocket attribute), 48
session_id (discordaio.voice.VoiceState attribute), 46
session_id (discordaio.websocket.DiscordWebsocket attribute), 48
shards (discordaio.websocket.DiscordWebsocket attribute), 48
size (discordaio.activity.ActivityParty attribute), 14
size (discordaio.channel.Attachment attribute), 26
small_image (discordaio.activity.ActivityAssets attribute), 15
small_text (discordaio.activity.ActivityAssets attribute), 15
splash (discordaio.guild.Guild attribute), 35
start (discordaio.activity.ActivityTimestamps attribute), 14
start() (discordaio.websocket.DiscordWebsocket method), 49
state (discordaio.activity.Activity attribute), 13
suppress (discordaio.voice.VoiceState attribute), 46
synced_at (discordaio.guild.Integration attribute), 39
syncing (discordaio.guild.Integration attribute), 39
system_channel_id (discordaio.guild.Guild attribute), 36

T

temporary (discordaio.invite.InviteMetadata attribute), 42
text (discordaio.channel.EmbedFooter attribute), 25
thumbnail (discordaio.channel.Embed attribute), 22
timestamp (discordaio.channel.ChannelMessage attribute), 18
timestamp (discordaio.channel.Embed attribute), 22
timestamps (discordaio.activity.Activity attribute), 13
title (discordaio.channel.Embed attribute), 21
token (discordaio.client.DiscordBot attribute), 27
token (discordaio.webhook.Webhook attribute), 47
topic (discordaio.channel.Channel attribute), 17
tts (discordaio.channel.ChannelMessage attribute), 18
type (discordaio.activity.Activity attribute), 13
type (discordaio.channel.ChannelMessage attribute), 19
type (discordaio.channel.Embed attribute), 21
type (discordaio.channel.MessageActivity attribute), 20
type (discordaio.channel.Overwrite attribute), 19
type (discordaio.guild.Integration attribute), 39
type (discordaio.user.UserConnection attribute), 44

U

unavailable (discordaio.guild.Guild attribute), 36
url (discordaio.activity.Activity attribute), 13
url (discordaio.channel.Attachment attribute), 26
url (discordaio.channel.Embed attribute), 21
url (discordaio.channel.EmbedAuthor attribute), 24
url (discordaio.channel.EmbedImage attribute), 23
url (discordaio.channel.EmbedProvider attribute), 24
url (discordaio.channel.EmbedThumbnail attribute), 22
url (discordaio.channel.EmbedVideo attribute), 23
User (class in discordaio.user), 43
user (discordaio.client.DiscordBot attribute), 27
user (discordaio.emoji.Emoji attribute), 30
user (discordaio.guild.Ban attribute), 39
user (discordaio.guild.GuildMember attribute), 37
user (discordaio.guild.Integration attribute), 39
user (discordaio.webhook.Webhook attribute), 47
user_id (discordaio.voice.VoiceState attribute), 46
user_limit (discordaio.channel.Channel attribute), 17
UserConnection (class in discordaio.user), 44
username (discordaio.user.User attribute), 43
uses (discordaio.invite.InviteMetadata attribute), 42

V

value (discordaio.channel.EmbedField attribute), 25
value_type (discordaio.channel.Channel attribute), 17
verification_level (discordaio.guild.Guild attribute), 35

VerificationLevel (class in discordaio.enums), 32
verified (discordaio.user.User attribute), 44
video (discordaio.channel.Embed attribute), 22
vip (discordaio.voice.VoiceRegion attribute), 46
voice_states (discordaio.guild.Guild attribute), 36
VoiceRegion (class in discordaio.voice), 46
VoiceState (class in discordaio.voice), 45

W

Webhook (class in discordaio.webhook), 47
webhook_id (discordaio.channel.ChannelMessage attribute), 19
WebSocketCreationError, 33
widget_channel_id (discordaio.guild.Guild attribute), 36
widget_enabled (discordaio.guild.Guild attribute), 36
width (discordaio.channel.Attachment attribute), 26
width (discordaio.channel.EmbedImage attribute), 23
width (discordaio.channel.EmbedThumbnail attribute), 22
width (discordaio.channel.EmbedVideo attribute), 23
ws (discordaio.client.DiscordBot attribute), 27