python-bale-bot

Release 2.5.0

Kian Ahmadian

Jan 22, 2024

REFERENCE

An API wrapper for Bale written in Python.

CHAPTER

ONE

INTRODUCTION

1.1 What is Bale?

The "Bale" is a messenger-platform for send and receive messages. it's provides services for developers, and they can send or receive messages through `bots` like normal users and These services are provided by web services (`API`).

1.2 What is python-bale-bot?

The "python-bale-bot" is a Python language package optimized for developers to use web services provided by "Bale".

CHAPTER

TWO

INSTALLING

You can install or update ``python-bale-bot`` via:

2.1 PyPi:

```
$ pip install python-bale-bot -U
```

2.2 Git:

```
$ git clone https://github.com/python-bale-bot/python-bale-bot
$ cd python-bale-bot
$ python setup.py install
```

CHAPTER

THREE

QUICK START

To get started, learn how the library works through the library. In addition, there are examples in the "Examples" section of the library.

CHAPTER

FOUR

DOCUMENTATION

The package documentation **is the technical reference for python-bale-bot. It contains descriptions of all available classes, modules, methods and arguments as well as the changelog.**

CHAPTER

FIVE

CONTACT TO DEVELOPERS

5.1 bale package

5.1.1 Classes in this package

Bot

class bale.**Bot**(*token*, ***kwargs*)

Bases: object

This object represents a Bale Bot.

Parameters
token (*str*) – Bot Token

Attention: When you create a bot and run for first-step, use *bale.Bot.delete_webhook()* method in *on_before_ready* event.

Examples

```
My First Bot
```

async ban_chat_member(*chat_id*, *user_id*)

Use this method to ban a user from a group, supergroup or a channel. In the case of supergroups and channels, the user will not be able to return to the group on their own using invite links, etc., unless unbanned first.

```
await bot.ban_chat_member(1234, 1234)
```

Parameters

- **chat_id** (Union[int, str]) – Unique identifier for the target chat or username of the target channel (in the format @channelusername).
- **user_id** (Union[int, str]) – Unique identifier of the target user.

Returns
On success, `True` is returned.

Return type
`bool`

Raises

- **NotFound** – Invalid Chat or User ID.
- **Forbidden** – You do not have permission to ban Chat Member.
- **APIError** – ban chat member Failed.

`property chats`

Represents the chats that the bot has ever encountered.

Type
`weakref.WeakValueDictionary`[:class:`str, *bale.Chat*]

`async close()`

Close http Events and bot

`async delete_message(`*chat_id*, *message_id*, *, *delay=None*`)`

You can use this service to delete a message that you have already sent through the arm.

```
await bot.delete_message(1234, 1234)
```

Warning: In channels or groups, only when the admin can delete other people's messages. Otherwise, It's no limit to delete his own message.

Parameters

- **chat_id** (Union[`str`, `int`]) – Unique identifier for the target chat or username of the target channel (in the format @channelusername).
- **message_id** (*bale.Message*) – Unique identifier for the message to delete.
- **delay** (Optional[Union[`int`, `float`]]) – If used, the message will be deleted after that number of seconds delay.

Raises

- **NotFound** – Invalid Message or Chat ID.
- **Forbidden** – You do not have permission to Delete Message.
- **APIError** – Delete Message Failed.

`async delete_webhook()`

This service is used to remove the webhook set for the bot.

```
await bot.delete_webhook()
```

Returns
On success, True is returned.

Return type
`bool`

Raises

- **Forbidden** – You do not have permission to delete Webhook.
- **APIError** – Delete webhook Failed.

async edit_message(*chat_id*, *message_id*, *text*, ***, *components=None*)

You can use this service to edit text messages that you have already sent through the arm.

```
await bot.edit_message(1234, 1234, "this is tested", components=None)
```

Parameters

- **chat_id** (Union[str, int]) – Unique identifier for the target chat or username of the target channel (in the format @channelusername).
- **message_id** (Union[str, int]) – Unique identifier for the message to edit.
- **text** (*str*) – New text of the message, 1- 4096 characters after entities parsing.
- **components** (Optional[Union[*bale.InlineKeyboardMarkup*, *bale.MenuKeyboardMarkup*]]) – An object for an inline keyboard.

Raises

- **NotFound** – Invalid Message or Chat ID.
- **Forbidden** – You do not have permission to Edit Message.
- **APIError** – Edit Message Failed.

event(*coro*)

Set wrapper or listener for selected event (the name of function).

```
@bot.event
async def on_message(message: bale.Message):
    ...
```

Hint: The name of the function for which you write the decorator is considered the name of the event.

async forward_message(*chat_id*, *from_chat_id*, *message_id*)

This service is used to send text messages.

```
await bot.forward_message(1234, 1234, 1234)
```

Parameters

- **chat_id** (Union[str, int]) – Unique identifier for the target chat or username of the target channel (in the format @channelusername).
- **from_chat_id** (Union[str, int]) – the chat where the original message was sent (or channel username in the format @channelusername).
- **message_id** (Union[int, str]) – Message in the chat specified in from_chat_id.

Returns

The Message

Return type
bale.Message

Raises

- **NotFound** – Invalid Chat ID.
- **Forbidden** – You do not have permission to send Message to this chat.
- **APIError** – Forward Message Failed.

async **get_chat**(*chat_id*, *, *use_cache=True*)

Use this method to get up-to-date information about the chat (current name of the user for one-on-one conversations, current username of a user, group or channel, etc.).

```
await bot.get_chat(1234)
...
await bot.get_chat("1234")
```

Parameters

- **chat_id** (Union[int, str]) – Unique identifier for the target chat or username of the target channel (in the format @channelusername).
- **use_cache** (Optional[bool]) – Use of caches stored in relation to chats.

Returns
The chat or None if not found.

Return type
Optional[*bale.Chat*]

Raises

- **Forbidden** – You do not have permission to get Chat.
- **APIError** – Get chat Failed.

async **get_chat_administrators**(*chat_id*)

Use this method to get a list of administrators in a chat.

```
await bot.get_chat_administrators(1234)
```

Parameters
chat_id (Union[str, int]) – Unique identifier for the target chat or username of the target channel (in the format @channelusername).

Returns
list of chat member.

Return type
List[*bale.ChatMember*]

Raises

- **NotFound** – Invalid Chat ID.
- **Forbidden** – You do not have permission to get Administrators of the Chat.
- **APIError** – get Administrators of the Chat from chat Failed.

async get_chat_member(*chat_id*, *user_id*)

Use this method to get information about a member of a chat. The method is only guaranteed to work for other users if the bot is an administrator in the chat.

```
await bot.get_chat_member(1234, 1234)
```

> **Warning:** Just only when the admin can ban member(s).

Parameters

- **chat_id** (Union[int, str]) – Unique identifier for the target chat or username of the target channel (in the format @channelusername).
- **user_id** (Union[int, str]) – Unique identifier of the target user.

Returns
The chat member of None if not found.

Return type
Optional[*bale.ChatMember*]

Raises

- **NotFound** – Invalid Chat or User ID.
- **Forbidden** – You do not have permission to get Chat Member.
- **APIError** – Get chat member Failed.

async get_chat_members_count(*chat_id*)

Use this method to get the number of members in a chat.

```
await bot.get_chat_members_count(1234)
```

Parameters
chat_id (Union[str, int]) – Unique identifier for the target chat or username of the target channel (in the format @channelusername).

Raises

- **NotFound** – Invalid Chat ID.
- **Forbidden** – You do not have permission to get Members count of the Chat.
- **APIError** – get Members count of the Chat Failed.

Returns
The members count of the chat

Return type
int

async get_file(*file_id*)

Use this method to get basic info about a file and prepare it for downloading. For the moment, bots can download files of up to 20 MB in size.

```
await bot.get_file("FILE_ID")
```

Parameters
file_id (str) – Either the file identifier to get file information about.

Returns
The content of the file

Return type
bytes

Raises

- **NotFound** – Invalid file ID.
- **Forbidden** – You do not have permission to download File.
- **APIError** – download File Failed.

async get_me()

Get bot information

Returns
Bot User information.

Return type
bale.User

Raises
APIError – Get bot Failed.

async get_user(*user_id*, *, *use_cache=True*)

This Method almost like *bale.Bot.get_chat()* , but this a filter that only get user.

```
await bot.get_user(1234)
...
await bot.get_user("1234")
```

Parameters

- **user_id** (Union[int, str]) – Unique identifier for the target chat.
- **use_cache** (Optional[bool]) – Use of caches stored in relation to chats.

Returns
The user or None if not found.

Return type
Optional[*bale.User*]

Raises

- **Forbidden** – You do not have permission to get User.
- **APIError** – Get user Failed.

http_is_closed()

bool: HTTPClient Status

async invite_user(*chat_id*, *user_id*)

Invite user to the chat

```
await bot.get_chat(1234, 1234)
```

Parameters

- **chat_id** (Union[str, int]) – Unique identifier for the target chat or username of the target channel (in the format @channelusername).
- **user_id** (Union[str, int]) – Unique identifier for the target user.

Raises

- **NotFound** – Invalid Chat or User ID.
- **Forbidden** – You do not have permission to Add user to Chat.
- **APIError** – Invite user Failed.

is_closed()

bool: Bot Status

async leave_chat(*chat_id*)

Use this method for your bot to leave a group, channel.

```
await bot.leave_chat(1234)
```

Parameters

chat_id (Union[str, int]) – Unique identifier for the target chat or username of the target channel (in the format @channelusername).

Raises

- **Forbidden** – You do not have permission to Leave from chat.
- **APIError** – Leave from chat Failed.

listen(*event_name*)

Set wrapper or listener for selected event (custom function name).

```
@bot.listen("on_message")
async def _message(message: bale.Message):
    ...
```

Parameters

event_name (str) – Name of the event to set.

async on_error(*event_name*, *error*)

an Event for get errors when exceptions

async promote_chat_member(*chat_id*, *user_id*, *can_be_edited=None*, *can_change_info=None*, *can_post_messages=None*, *can_edit_messages=None*, *can_delete_messages=None*, *can_invite_users=None*, *can_restrict_members=None*, *can_pin_messages=None*, *can_promote_members=None*, *can_send_messages=None*, *can_send_media_messages=None*, *can_reply_to_story=None*, *can_send_link_message=None*, *can_send_forwarded_message=None*, *can_see_members=None*, *can_add_story=None*)

an administrator in the chat for this to work and must have the appropriate admin rights. Pass False for all boolean parameters to demote a user.

```
await bot.promote_chat_member(1234, 1234, can_change_info = True)
```

Parameters

- **chat_id** (Union[int, str]) – Unique identifier for the target chat or username of the target channel (in the format @channelusername).
- **user_id** (Union[int, str]) – Unique identifier of the target user.
- **can_be_edited** (bool) – Pass True, if the bot is allowed to edit administrator privileges of that user.
- **can_change_info** (bool) – Pass True, if the user can change the chat title, photo and other settings.
- **can_post_messages** (bool) – Pass True, if the administrator can post messages in the channel, or access channel statistics; channels only.
- **can_edit_messages** (bool) – Pass True, if the administrator can edit messages of other users and can pin messages; channels only.
- **can_delete_messages** (bool) – Pass True, if the administrator can delete messages of other users.
- **can_invite_users** (bool) – Pass True, if the user can invite new users to the chat.
- **can_restrict_members** (bool) – Pass True, if the administrator can restrict, ban or unban chat members.
- **can_pin_messages** (bool) – Pass True, if the user is allowed to pin messages, groups, channels only.
- **can_promote_members** (bool) – Pass True, if the administrator can add new administrators with a subset of his own privileges or demote administrators that he has promoted, directly or indirectly (promoted by administrators that were appointed by the user).
- **can_send_messages** (bool) – Pass True, if the user is allowed to send messages.
- **can_send_media_messages** (bool) – Pass True, if the user is allowed to send a media message.
- **can_reply_to_story** (bool) – Pass True, if the user is allowed to reply to a story.
- **can_send_link_message** (bool) – Pass True, if the user is allowed to send a link message.
- **can_send_forwarded_message** (bool) – Pass True, if the user is allowed to forward a message to chat.
- **can_see_members** (bool) – Pass True, if the user is allowed to see the list of chat members.
- **can_add_story** (bool) – Pass True, if the user is allowed to post a story from chat.

Returns

On success, True is returned.

Return type

bool

Raises

- **NotFound** – Invalid Chat or User ID.

- **Forbidden** – You do not have permission to promote Chat Member.
- **APIError** – Promote chat member Failed.

run()

Starting the bot, updater and HTTPClient.

async send_animation(*chat_id*, *animation*, ***, *duration=None*, *width=None*, *height=None*, *caption=None*, *components=None*, *reply_to_message_id=None*, *delete_after=None*)

This service is used to send Animation.

```
await bot.send_animation(1234, bale.InputFile("FILE_ID"), caption = "this is a␣
↪caption", ...)
```

Parameters

- **chat_id** (Union[str, int]) – Unique identifier for the target chat or username of the target channel (in the format @channelusername).
- **animation** (*bale.InputFile*) – File to send. visit *bale.InputFile* to see more info.
- **duration** (int) – Duration of sent animation in seconds.
- **width** (int) – Animation width.
- **height** (int) – Animation height.
- **caption** (Optional[str]) – Animation caption.
- **components** (Optional[Union[*bale.InlineKeyboardMarkup*, *bale.MenuKeyboardMarkup*]]) – Message Components
- **reply_to_message_id** (Optional[Union[str, int]]) – If the message is a reply, ID of the original message.
- **delete_after** (Optional[Union[float, int]]) – If used, the sent message will be deleted after the specified number of seconds.

Returns

The Message.

Return type

bale.Message

Raises

- **NotFound** – Invalid Chat ID.
- **Forbidden** – You do not have permission to Send Animation to chat.
- **APIError** – Send Animation Failed.

async send_audio(*chat_id*, *audio*, ***, *caption=None*, *components=None*, *reply_to_message_id=None*, *delete_after=None*)

This service is used to send Audio.

```
await bot.send_audio(1234, bale.InputFile("FILE_ID"), caption = "this is a␣
↪caption", ...)
```

Parameters

- **chat_id** (Union[str, int]) – Unique identifier for the target chat or username of the target channel (in the format @channelusername).
- **audio** (*bale.InputFile*) – File to send. visit *bale.InputFile* to see more info.
- **caption** (Optional[str]) – Audio caption.
- **components** (Optional[Union[*bale.InlineKeyboardMarkup*, *bale.MenuKeyboardMarkup*]]) – Message Components
- **reply_to_message_id** (Optional[Union[str, int]]) – If the message is a reply, ID of the original message.
- **delete_after** (Optional[Union[float, int]]) – If used, the sent message will be deleted after the specified number of seconds.

Returns
The Message.

Return type
bale.Message

Raises

- **NotFound** – Invalid Chat ID.
- **Forbidden** – You do not have permission to Send Audio to chat.
- **APIError** – Send Audio Failed.

async send_contact(*chat_id*, *contact*, *components=None*, *reply_to_message_id=None*, *delete_after=None*)

This service is used to send contact.

```
await bot.send_cantact(1234, bale.Contact('09****', 'first name', 'last name'))
```

Parameters

- **chat_id** (Union[str, int]) – Unique identifier for the target chat or username of the target channel (in the format @channelusername).
- **contact** (*bale.Contact*) – The Contact.
- **components** (Optional[Union[*bale.InlineKeyboardMarkup*, *bale.MenuKeyboardMarkup*]]) – Message Components
- **reply_to_message_id** (Optional[Union[str, int]]) – If the message is a reply, ID of the original message.
- **delete_after** (Optional[Union[float, int]]) – If used, the sent message will be deleted after the specified number of seconds.

Returns
The Message.

Return type
bale.Message

Raises

- **NotFound** – Invalid Chat ID.
- **Forbidden** – You do not have permission to send Contact Message to this chat.

- **APIError** – Send Contact Message Failed.

async send_document(*chat_id*, *document*, ***, *caption=None*, *components=None*, *reply_to_message_id=None*, *delete_after=None*)

This service is used to send document.

```
await bot.send_document(1234, bale.InputFile("FILE_ID"), caption = "this is a␣
↪caption", ...)
```

Parameters

- **chat_id** (Union[str, int]) – Unique identifier for the target chat or username of the target channel (in the format @channelusername).
- **document** (*bale.InputFile*) – File to send. visit *bale.InputFile* to see more info.
- **caption** (Optional[str]) – Document caption.
- **components** (Optional[Union[*bale.InlineKeyboardMarkup*, *bale.MenuKeyboardMarkup*]]) – Message Components
- **reply_to_message_id** (Optional[Union[str, int]]) – If the message is a reply, ID of the original message.
- **delete_after** (Optional[Union[float, int]]) – If used, the sent message will be deleted after the specified number of seconds.

Returns
The Message.

Return type
bale.Message

Raises

- **NotFound** – Invalid Chat ID.
- **Forbidden** – You do not have permission to send Document to this chat.
- **APIError** – Send Document Failed.

async send_invoice(*chat_id*, *title*, *description*, *provider_token*, *prices*, ***, *payload=None*, *photo_url=None*, *need_name=False*, *need_phone_number=False*, *need_email=False*, *need_shipping_address=False*, *is_flexible=True*, *delete_after=None*)

You can use this service to send money request messages.

Important: When paying the amount, a fee will be charged from the sender.

Hint: The *on_successful_payment* event is called when the sent transaction is done.

```
await bot.send_invoice(
    1234, "invoice title", "invoice description", "6037************", [bale.
↪LabeledPrice("label", 2000)],
    payload = "unique invoice payload", ...
)
```

Examples

```
Payment Bot
```

Parameters

- **chat_id** (Union[str, int]) – Unique identifier for the target chat or username of the target channel (in the format @channelusername).
- **title** (*str*) – Product name. 1- 32 characters.
- **description** (*str*) – Product description. 1- 255 characters.
- **provider_token** (*str*) – You can use 3 methods to receive money: 1.Card number 2. Port number and acceptor number 3. Wallet number "Bale"
- **prices** (List[*bale.LabeledPrice*]) – A list of prices.
- **payload** (Optional[str]) – Bot-defined invoice payload. This will not be displayed to the user, use for your internal processes.
- **photo_url** (Optional[str]) – URL of the product photo for the invoice. Can be a photo of the goods or a marketing image for a service. People like it better when they see what they are paying for.
- **need_name** (*Optional[bool]*) – Pass True, if you require the user's full name to complete the order.
- **need_phone_number** (*Optional[bool]*) – Pass True, if you require the user's phone number to complete the order.
- **need_email** (*Optional[bool]*) – Pass True, if you require the user's email to complete the order.
- **need_shipping_address** (*Optional[bool]*) – Pass True, if you require the user's shipping address to complete the order.
- **is_flexible** (*Optional[bool]*) – Pass True, if the final price depends on the shipping method.
- **delete_after** (Optional[Union[float, int]]) – If used, the sent message will be deleted after the specified number of seconds.

Return type
bale.Message

Raises

- **NotFound** – Invalid Chat ID.
- **Forbidden** – You do not have permission to send Invoice to this chat.
- **APIError** – Send Invoice Failed.

async send_location(*chat_id*, *location*, *components=None*, *reply_to_message_id=None*, *delete_after=None*)

Use this method to send point on the map.

```
await bot.send_location(1234, bale.Location(35.71470468031143, 51.
↪8568519168293))
```

Parameters

- **chat_id** (Union[str, int]) – Unique identifier for the target chat or username of the target channel (in the format @channelusername).
- **location** (*bale.Location*) – The Location.
- **components** (Optional[Union[*bale.InlineKeyboardMarkup*, *bale.MenuKeyboardMarkup*]]) – Message Components
- **reply_to_message_id** (Optional[Union[str, int]]) – If the message is a reply, ID of the original message.
- **delete_after** (Optional[Union[float, int]]) – If used, the sent message will be deleted after the specified number of seconds.

Returns
The Message.

Return type
bale.Message

Raises

- **NotFound** – Invalid Chat ID.
- **Forbidden** – You do not have permission to send Location to this chat.
- **APIError** – Send Location Failed.

async send_message(*chat_id*, *text*, ***, *components=None*, *reply_to_message_id=None*, *delete_after=None*)

This service is used to send text messages.

```
await bot.send_message(1234, "hi, python-bale-bot!", ...)
```

Parameters

- **chat_id** (Union[str, int]) – Unique identifier for the target chat or username of the target channel (in the format @channelusername).
- **text** (str) – Text of the message to be sent. Max 4096 characters after entities parsing.
- **components** (Optional[Union[*bale.InlineKeyboardMarkup*, *bale.MenuKeyboardMarkup*]]) – Message Components
- **reply_to_message_id** (Optional[Union[str, int]]) – If the message is a reply, ID of the original message.
- **delete_after** (Optional[Union[float, int]]) – If used, the sent message will be deleted after the specified number of seconds.

Returns
The Message

Return type
bale.Message

Raises

- **NotFound** – Invalid Chat ID.
- **Forbidden** – You do not have permission to send Message to this chat.
- **APIError** – Send Message Failed.

async send_photo(*chat_id*, *photo*, ***, *caption=None*, *components=None*, *reply_to_message_id=None*, *delete_after=None*)

This service is used to send photo.

```
await bot.send_photo(1234, bale.InputFile("FILE_ID"), caption = "this is a␣
↪caption", ...)
```

Parameters

- **chat_id** (Union[str, int]) – Unique identifier for the target chat or username of the target channel (in the format @channelusername).
- **photo** (*bale.InputFile*) – File to send. visit *bale.InputFile* to see more info.
- **caption** (Optional[str]) – Photo caption.
- **components** (Optional[Union[*bale.InlineKeyboardMarkup*, *bale.MenuKeyboardMarkup*]]) – Message Components
- **reply_to_message_id** (Optional[Union[str, int]]) – If the message is a reply, ID of the original message.
- **delete_after** (Optional[Union[float, int]]) – If used, the sent message will be deleted after the specified number of seconds.

Returns
The Message.

Return type
bale.Message

Raises

- **NotFound** – Invalid Chat ID.
- **Forbidden** – You do not have permission to Send Photo to chat.
- **APIError** – Send photo Failed.

async send_video(*chat_id*, *video*, ***, *caption=None*, *components=None*, *reply_to_message_id=None*, *delete_after=None*)

This service is used to send Video.

```
await bot.send_video(1234, bale.InputFile("FILE_ID"), caption = "this is a␣
↪caption", ...)
```

Parameters

- **chat_id** (Union[str, int]) – Unique identifier for the target chat or username of the target channel (in the format @channelusername).
- **video** (*bale.InputFile*) – File to send. visit *bale.InputFile* to see more info.
- **caption** (Optional[str]) – Video caption.
- **components** (Optional[Union[*bale.InlineKeyboardMarkup*, *bale.MenuKeyboardMarkup*]]) – Message Components
- **reply_to_message_id** (Optional[Union[str, int]]) – If the message is a reply, ID of the original message.

- **delete_after** (Optional[Union[float, int]]) – If used, the sent message will be deleted after the specified number of seconds.

Returns
The Message.

Return type
bale.Message

Raises

- **NotFound** – Invalid Chat ID.
- **Forbidden** – You do not have permission to Send Video to chat.
- **APIError** – Send Video Failed.

async set_chat_photo(*chat_id*, *photo*)

Use this method to set a new profile photo for the chat.

```
await bot.set_chat_photo(1234, bale.InputFile("FILE_ID"))
```

Parameters

- **chat_id** (Union[str, int]) – Unique identifier for the target chat or username of the target channel (in the format @channelusername).
- **photo** (*bale.InputFile*) – New chat photo. visit *bale.InputFile* to see more info.

Returns
On success, True is returned.

Return type
bool

Raises

- **NotFound** – Invalid Chat ID.
- **Forbidden** – You do not have permission to Set Chat Photo to chat.
- **APIError** – Set chat photo Failed.

async set_webhook(*url*)

Use this method to specify an url and receive incoming updates via an outgoing webhook. Whenever there is an update for the bot, Bale will send an HTTPS POST request to the specified url, containing An Update. In case of an unsuccessful request, Bale will give up after a reasonable amount of attempts.

```
await bot.set_webhook("https://example.com")
```

Parameters
url (str) – HTTPS url to send updates to. Use an empty string to remove webhook integration.

Returns
On success, True is returned.

Return type
bool

property state

Represents the state class for cache data. None if bot not logged in

Type
Optional[bale.State]

async unban_chat_member(*chat_id*, *user_id*, *, *only_if_banned=None*)

Use this method to unban a previously kicked user in a group or channel. The user will not return to the group or channel automatically, but will be able to join via link, etc. The bot must be an administrator for this to work. By default, this method guarantees that after the call the user is not a member of the chat, but will be able to join it. So if the user is a member of the chat they will also be removed from the chat. If you don't want this, use the parameter only_if_banned.

```
await bot.unban_chat_member(1234, 1234)
```

Parameters

- **chat_id** (Union[int, str]) – Unique identifier for the target chat or username of the target channel (in the format @channelusername).
- **user_id** (Union[int, str]) – Unique identifier of the target user.
- **only_if_banned** (Optional[bool]) – Do nothing if the user is not banned.

Returns
On success, True is returned.

Return type
bool

Raises

- **NotFound** – Invalid Chat or User ID.
- **Forbidden** – You do not have permission to unban Chat Member.
- **APIError** – unban chat member Failed.

property user

Represents the connected client. None if not logged in

Type
Optional[*bale.User*]

property users

Represents the users that the bot has ever encountered.

Type
weakref.WeakValueDictionary`[:class:`str, *bale.User*]

wait_for(*event_name*, *, *check=None*, *timeout=None*)

Waits for an event to be dispatched.

This could be used to wait for a user to reply to a message, or send a photo, or to edit a message in a self-contained way. The timeout parameter is passed onto asyncio.wait_for(). By default, it does not timeout. Note that this does propagate the asyncio.TimeoutError for you in case of timeout and is provided for ease of use. In case the event returns multiple arguments, a tuple containing those arguments is returned instead. This function returns the first event that meets the requirements.

```
message = await bot.wait_for("message", check = lambda m: m.author.user_id ==
↪'1234')
...
try:
    message = await bot.wait_for("message", ..., timeout = 20.0)
except asyncio.TimeoutError: # 20s A message with the conditions specified in␣
↪the `check` parameter was not found.
    pass
```

Examples

conversation Bot

Parameters

- **event_name** (str) – Name of the event
- **check** (Optional[Callable[..., bool]]) – A predicate to check what to wait for. The arguments must meet the parameters of the event being waited for.
- **timeout** (Optional[float]) – The number of seconds to wait before timing out and raising asyncio.TimeoutError.

Raises

asyncio.TimeoutError – If a timeout is provided, and it was reached.

Available Types

Update

class bale.**Update**(*update_id*, *callback_query=None*, *message=None*, *edited_message=None*)

Bases: BaleObject

This object represents an incoming update.

update_id

The update's unique identifier. Update identifiers start from a certain positive number and increase sequentially. This ID becomes especially handy if you're using Webhooks, since it allows you to ignore repeated updates or to restore the correct update sequence, should they get out of order. If there are no new updates for at least a week, then identifier of the next update will be chosen randomly instead of sequentially.

Type

int

callback_query

New incoming callback query.

Type

Optional[*bale.CallbackQuery*]

message

New incoming message of any kind - text, photo, sticker, etc.

Type

Optional[*bale.Message*]

edited_message

New version of a message that is known to the bot and was edited.

Type
Optional[*bale.Message*]

Message

class bale.**Message**(*message_id*, *date*, *text*, *caption*, *forward_from*, *forward_from_chat*, *forward_from_message_id*, *from_user*, *document*, *contact*, *edit_date*, *location*, *chat*, *video*, *photos*, *sticker*, *reply_to_message*, *invoice*, *audio*, *successful_payment*, *animation*, *new_chat_members*, *left_chat_member*)

Bases: BaleObject

This object shows a message.

message_id

Unique message identifier inside this chat.

Type
str

from_user

Sender of the message; empty for messages sent to channels. For backward compatibility, this will contain a fake sender user in non-channel chats, if the message was sent on behalf of a chat.

Type
Optional[*bale.User*]

chat

Conversation the message belongs to.

Type
bale.Chat

date

Date the message was sent in Unix time.

Type
datetime.datetime

text

Message Content

Type
Optional[str]

caption

Caption for the animation, audio, document, photo, video or voice.

Type
Optional[str]

forward_from

For forwarded messages, sender of the original message.

Type
Optional[*bale.User*]

forward_from_chat

For messages forwarded from channels or from anonymous administrators, information about the original sender chat.

Type
Optional[*bale.Chat*]

reply_to_message

For replies, the original message. Note that the Message object in this field will not contain further reply_to_message fields even if it itself is a reply.

Type
Optional[*bale.Message*]

edit_date

Date the message was last edited.

Type
Optional[datetime.datetime]

contact

Message is a shared contact, information about the contact.

Type
Optional[*bale.Contact*]

location

Message is a shared location, information about the location.

Type
Optional[*bale.Location*]

document

Message is a general file, information about the file.

Type
Optional[*bale.Document*]

video

Message is a video, information about the video.

Type
Optional[*bale.Video*]

animation

Message is an animation, information about the animation.

Type
Optional[*bale.Animation*]

audio

Message is an audio, information about the Audio.

Type
Optional[*bale.Audio*]

sticker

Message is a sticker, information about the sticker.

Type
Optional[*bale.Sticker*]

new_chat_members

New members that were added to the group or supergroup and information about them (the bot itself may be one of these members). This list is empty if the message does not contain new chat members.

Type
Optional[List[*bale.User*]]

left_chat_member

A member was removed from the group, information about them (this member may be the bot itself).

Type
Optional[*bale.User*]

invoice

Message is an invoice for a payment, information about the invoice.

Type
Optional[*bale.Invoice*]

successful_payment

Message is a service message about a successful payment, information about the payment.

Type
Optional[*bale.SuccessfulPayment*]

property attachment

Represents the message attachment. `None` if the message don't have any attachments

Type
Optional[*bale.BaseFile*]

property author

An alias for *from_user*

property chat_id

Represents the Unique identifier of Conversation the message belongs to.

Type
`str` | `int`

property content

Represents the message content. `None` if the message don't have text or caption

Type
Optional[`str`]

async delete(*, *delay=None*)

For the documentation of the arguments, please see *bale.Bot.delete_message()*.

```
await message.delete(delay=5)
```

async edit(*text*, *, *components=None*)

For the documentation of the arguments, please see *bale.Bot.edit_message()*.

```
await message.edit("Bye!", components = None)
```

async forward(*chat_id*)

For the documentation of the arguments, please see *bale.Bot.forward_message()*.

```
await message.forward(1234)
```

async reply(*text*, ***, *components=None*, *delete_after=None*)

For the documentation of the arguments, please see *bale.Bot.send_message()*.

```
await message.reply("Hi, python-bale-bot!", components = None)
```

async reply_animation(*animation*, ***, *duration=None*, *width=None*, *height=None*, *caption=None*, *components=None*, *delete_after=None*)

For the documentation of the arguments, please see *bale.Bot.send_animation()*.

```
await message.reply_animation(bale.InputFile("FILE_ID"), caption = "this is a␣
↪caption", ...)
```

async reply_audio(*audio*, ***, *caption=None*, *components=None*, *delete_after=None*)

For the documentation of the arguments, please see *bale.Bot.send_audio()*.

```
await message.reply_audio(bale.InputFile("FILE_ID"), caption = "this is a␣
↪caption", ...)
```

async reply_contact(*contact*, ***, *components=None*, *delete_after=None*)

For the documentation of the arguments, please see *bale.Bot.send_contact()*.

```
await message.reply_contact(bale.Contact('09****', 'first name', 'last name'))
```

async reply_document(*document*, ***, *caption=None*, *components=None*, *delete_after=None*)

For the documentation of the arguments, please see *bale.Bot.send_document()*.

```
await message.reply_document(bale.InputFile("FILE_ID"), caption = "this is a␣
↪caption", ...)
```

async reply_location(*location*, ***, *components=None*, *delete_after=None*)

For the documentation of the arguments, please see *bale.Bot.send_audio()*.

```
await message.reply_location(bale.Location(35.71470468031143, 51.8568519168293))
```

async reply_photo(*photo*, ***, *caption=None*, *components=None*, *delete_after=None*)

For the documentation of the arguments, please see *bale.Bot.send_photo()*.

```
await message.reply_photo(bale.InputFile("FILE_ID"), caption = "this is a␣
↪caption", ...)
```

property reply_to_message_id

Represents the Unique identifier of Original message, if the message has been replied. None If the message is not replied

Type
Optional[str]

async reply_video(*video*, ***, *caption=None*, *components=None*, *delete_after=None*)

For the documentation of the arguments, please see *bale.Bot.send_video()*.

```
await message.reply_video(bale.InputFile("FILE_ID"), caption = "this is a␣
↪caption", ...)
```

Chat

class bale.Chat(*id*, *type*, *title*, *username*, *first_name*, *last_name*, *photo*, *invite_link*)

Bases: BaleObject

This object indicates a chat.

id

Unique identifier for this chat.

Type
str

type

Type of chat.

Type
str

title

Title, for channels and group chats.

Type
Optional[str]

username

Username, for private chats, supergroups and channels if available.

Type
Optional[str]

first_name

First name of the other party in a private chat.

Type
Optional[str]

last_name

Last name of the other party in a private chat.

Type
Optional[str]

photo

Chat photo.

Type
Optional[*bale.ChatPhoto*]

invite_link

Primary invite link, for groups and channel. Returned only in *bale.Bot.get_chat()*.

Type
Optional[str]

async add_user(*user*)

For the documentation of the arguments, please see *bale.Bot.invite_user()*.

```
user = await bot.get_user(1234)
await chat.add_user(user)
```

async ban_chat_member(*user*)

For the documentation of the arguments, please see *bale.Bot.ban_chat_member()*.

```
user = await bot.get_user(1234)
await chat.ban_chat_member(user)
...
await chat.ban_chat_member(1234)
```

async get_chat_administrators()

For the documentation of the arguments, please see *bale.Bot.get_chat_administrators()*.

```
await chat.get_chat_administrators()
```

async get_chat_member(*user*)

For the documentation of the arguments, please see *bale.Bot.get_chat_member()*.

```
user = await bot.get_user(1234)
await chat.get_chat_member(user)
...
await chat.get_chat_member(1234)
```

async get_chat_members_count()

For the documentation of the arguments, please see *bale.Bot.get_chat_members_count()*.

```
await chat.get_chat_members_count()
```

async leave()

For the documentation of the method, please see *bale.Bot.leave_chat()*.

```
chat = await bot.get_chat(1234)
await chat.leave()
```

async send(*text*, *components=None*, *delete_after=None*)

For the documentation of the arguments, please see *bale.Bot.send_message()*.

```
await chat.send("hi, python-bale-bot!", components = None)
```

async send_animation(*animation*, ***, *duration=None*, *width=None*, *height=None*, *caption=None*, *components=None*, *delete_after=None*)

For the documentation of the arguments, please see *bale.Bot.send_animation()*.

```
await chat.send_animation(bale.InputFile("FILE_ID"), caption = "this is caption
↪", ...)
```

async send_audio(*audio*, ***, *caption=None*, *components=None*, *delete_after=None*)

For the documentation of the arguments, please see *bale.Bot.send_audio()*.

```
await chat.send_audio(bale.InputFile("FILE_ID"), caption = "this is caption", ..
↪.)
```

async send_contact(*contact*, *delete_after=None*)

For the documentation of the arguments, please see *bale.Bot.send_contact()*.

```
await chat.send_contact(Contact('09****', 'first name', 'last name))
```

async send_document(*document*, ***, *caption=None*, *components=None*, *delete_after=None*)

For the documentation of the arguments, please see *bale.Bot.send_document()*.

```
await chat.send_document(bale.InputFile("FILE_ID"), caption = "this is caption",
↪ ...)
```

async send_invoice(*title*, *description*, *provider_token*, *prices*, ***, *payload=None*, *photo_url=None*, *need_name=False*, *need_phone_number=False*, *need_email=False*, *need_shipping_address=False*, *is_flexible=True*, *delete_after=None*)

For the documentation of the arguments, please see *bale.Bot.send_invoice()*.

```
await chat.send_invoice(
    "invoice title", "invoice description", "6037************", [bale.
↪LabeledPrice("label", 2000)],
    payload = "unique invoice payload", ...
)
```

async send_location(*location*, *delete_after=None*)

For the documentation of the arguments, please see *bale.Bot.send_location()*.

```
await chat.send_location(bale.Location(35.71470468031143, 51.8568519168293))
```

async send_photo(*photo*, ***, *caption=None*, *components=None*, *delete_after=None*)

For the documentation of the arguments, please see *bale.Bot.send_photo()*.

```
await chat.send_photo(bale.InputFile("FILE_ID"), caption = "this is caption", ..
↪.)
```

async send_video(*video*, ***, *caption=None*, *components=None*, *delete_after=None*)

For the documentation of the arguments, please see *bale.Bot.send_video()*.

```
await chat.send_video(bale.InputFile("FILE_ID"), caption = "this is caption", ..
↪.)
```

async set_photo(*photo*)

For the documentation of the arguments, please see *bale.Bot.set_chat_photo()*.

```
await chat.set_photo(bale.InputFile("FILE_ID"))
```

async unban_chat_member(*user*, ***, *only_if_banned=None*)

For the documentation of the arguments, please see *bale.Bot.unban_chat_member()*.

```
user = await bot.get_user(1234)
await chat.unban_chat_member(user)
...
await chat.unban_chat_member(1234)
```

Chat Member

class bale.**ChatMember**(*status*, *user*, *is_member*, *can_change_info*, *can_post_messages*, *can_edit_messages*, *can_delete_messages*, *can_invite_users*, *can_restrict_members*, *can_pin_messages*, *can_promote_members*, *can_send_messages*, *can_send_media_messages*, *can_reply_to_story*, *can_send_link_message*, *can_send_forwarded_message*, *can_see_members*, *can_add_story*, *can_be_edited*)

Bases: `BaleObject`

This object shows a member in chat.

user

Information about the chat member.

Type
bale.User

status

The member's status in the chat.

Type
str

can_be_edited

`True`, if the bot is allowed to edit administrator privileges of that user.

Type
bool

can_change_info

`True`, if the user can change the chat title, photo and other settings.

Type
bool

can_post_messages

`True`, if the administrator can post messages in the channel, or access channel statistics; channels only.

Type
bool

can_edit_messages

`True`, if the administrator can edit messages of other users and can pin messages; channels only.

Type
bool

can_delete_messages

`True`, if the administrator can delete messages of other users.

Type
bool

can_invite_users

`True`, if the user can invite new users to the chat.

Type
bool

can_restrict_members

True, if the administrator can restrict, ban or unban chat members.

Type
bool

can_pin_messages

True, if the user is allowed to pin messages, groups, channels only.

Type
bool

can_promote_members

True, if the administrator can add new administrators with a subset of his own privileges or demote administrators that he has promoted, directly or indirectly (promoted by administrators that were appointed by the user).

Type
bool

can_send_messages

True, if the user is allowed to send messages.

Type
bool

can_send_media_messages

True, if the user is allowed to send a media message.

Type
bool

can_reply_to_story

True, if the user is allowed to reply to a story.

Type
bool

can_send_link_message

True, if the user is allowed to send a link message.

Type
bool

can_send_forwarded_message

True, if the user is allowed to forward a message to chat.

Type
bool

can_see_members

True, if the user is allowed to see the list of chat members.

Type
bool

can_add_story

True, if the user is allowed to post a story from chat.

Type
bool

Chat Photo

class bale.**ChatPhoto**(*small_file_id=None*, *small_file_unique_id=None*, *big_file_id=None*, *big_file_unique_id=None*)

Bases: `BaleObject`

This object represents a chat photo.

small_file_id

File identifier of small (160 x 160) chat photo.

Type
Optional[str]

small_file_unique_id

Unique file identifier of small (160 x 160) chat photo.

Type
Optional[str]

big_file_id

File identifier of big (640 x 640) chat photo.

Type
Optional[str]

big_file_unique_id

Unique file identifier of big (640 x 640) chat photo.

Type
Optional[str]

property **big_file_object**

Represents the big file object.

Type
Optional[*bale.BaseFile*]

property **small_file_object**

Represents the small file object.

Type
Optional[*bale.BaseFile*]

User

class bale.**User**(*id*, *is_bot*, *first_name*, *last_name=None*, *username=None*)

Bases: `BaleObject`

This object represents a Bale user or bot.

id

Unique identifier for this user or bot.

Type
int

is_bot

True, if this user is a bot.

Type
bool

first_name

User's or bot's first name.

Type
str

last_name

User's or bot's last name.

Type
Optional[str]

username

User's or bot's username.

Type
Optional[str]

property chat_id

Aliases for *id*

property mention

mention user with username.

Type
Optional[str]

async send(*text*, *components=None*, *delete_after=None*)

For the documentation of the arguments, please see *bale.Bot.send_message()*.

```
await user.send("Hi, python-bale-bot!", components = None)
```

async send_animation(*animation*, *, *duration=None*, *width=None*, *height=None*, *caption=None*, *components=None*, *delete_after=None*)

For the documentation of the arguments, please see *bale.Bot.send_animation()*.

```
await user.send_animation(bale.InputFile("FILE_ID"), caption = "this is a␣
↪caption", ...)
```

async send_audio(*audio*, *, *caption=None*, *components=None*, *delete_after=None*)

For the documentation of the arguments, please see *bale.Bot.send_audio()*.

```
await user.send_audio(bale.InputFile("FILE_ID"), caption = "this is a caption",␣
↪...)
```

async send_contact(*contact*, *, *components=None*, *delete_after=None*)

For the documentation of the arguments, please see *bale.Bot.send_contact()*.

```
await user.send_contact(bale.Contact('09****', 'first name', 'last name))
```

async send_document(*document*, ***, *caption=None*, *components=None*, *delete_after=None*)

For the documentation of the arguments, please see *bale.Bot.send_document()*.

```
await user.send_document(bale.InputFile("FILE_ID"), caption = "this is a caption
↪", ...)
```

async send_invoice(*title*, *description*, *provider_token*, *prices*, ***, *payload=None*, *photo_url=None*, *need_name=False*, *need_phone_number=False*, *need_email=False*, *need_shipping_address=False*, *is_flexible=True*, *delete_after=None*)

For the documentation of the arguments, please see *bale.Bot.send_invoice()*.

```
await user.send_invoice(
    "invoice title", "invoice description", "6037************", [bale.
↪LabeledPrice("label", 2000)],
    payload = "unique invoice payload", ...
)
```

async send_location(*location*, ***, *components=None*, *delete_after=None*)

For the documentation of the arguments, please see *bale.Bot.send_location()*.

```
await user.send_location(bale.Location(35.71470468031143, 51.8568519168293))
```

async send_photo(*photo*, ***, *caption=None*, *components=None*, *delete_after=None*)

For the documentation of the arguments, please see *bale.Bot.send_photo()*.

```
await user.send_photo(bale.InputFile("FILE_ID"), caption = "this is a caption",␣
↪...)
```

async send_video(*video*, ***, *caption=None*, *components=None*, *delete_after=None*)

For the documentation of the arguments, please see *bale.Bot.send_video()*.

```
await user.send_video(bale.InputFile("FILE_ID"), caption = "this is a caption",␣
↪...)
```

property user_id

Aliases for *id*

Callback Query

class bale.**CallbackQuery**(*id*, *from_user*, *data*, *message*, *inline_message_id*)

Bases: BaleObject

This object represents an incoming callback query from a callback button in an inline keyboard.

id

Unique identifier for this query.

Type
str

from_user

Sender.

Type
bale.User

message

Message with the callback button that originated the query. Note that message content and message date will not be available if the message is too old.

Type
Optional[*bale.Message*]

inline_message_id

Identifier of the message sent via the bot in inline mode, that originated the query.

Type
Optional[str]

data

Data associated with the callback button. Be aware that the message, which originated the query, can contain no callback buttons with this data.

Type
Optional[str]

property user

Aliases for *from_user*

Sticker

class bale.**Sticker**(*file_id*, *file_unique_id*, *type*, *width*, *height*, *thumb*, *set_name=None*, *file_size=None*)

Bases: BaleObject

This object shows a Sticker.

file_id

Identifier for this sticker file, which can be used to download or reuse the file.

Type
str

file_unique_id

Unique file identifier of sticker.

Type
str

type

Type of the sticker. Currently one of regular and mask.

Type
str

width

Sticker width.

Type
int

height

Sticker height.

Type
str

thumb

Sticker thumbnail.

Type
Optional[*bale.PhotoSize*]

set_name

Name of the sticker set to which the sticker belongs.

Type
Optional[str]

file_size

File size in bytes.

Type
Optional[int]

async get_file()

For the documentation of the arguments, please see *bale.Bot.get_file()*.

UI

Inline Keyboard Markup

class bale.**InlineKeyboardMarkup**

Bases: BaseReplyMarkup

Examples

Components Bot

---

add(*inline_keyboard_button*, *row=None*)

Add an Inline Keyboard button to keyboards.

Parameters

- **inline_keyboard_button** (*bale.InlineKeyboardButton*) – The inline keyboard button.
- **row** (Optional[int]) – The row where you want the button to be placed.

> **Warning:** Your numbers in the "row" param must be natural and greater than 0.

remove(*item*)

Remove a Reply Markup item from keyboards.

Parameters
item (*bale.ReplyMarkupItem*) – The reply markup item.

remove_row(*row*)

Remove a row along with the inline keyboards located in that row.

Parameters
row (int) – The row.

> **Warning:** Your numbers in the "row" param must be natural and greater than 0.

Menu Keyboard Markup

class bale.**MenuKeyboardMarkup**

Bases: BaseReplyMarkup

Examples

```
Components Bot
```

---

add(*keyboard_button*, *row=None*)

Add a Menu Keyboard button to keyboards.

Parameters

- **keyboard_button** (*bale.MenuKeyboardButton*) – The menu keyboard button.
- **row** (Optional[int]) – The row where you want the button to be placed.

> **Warning:** Your numbers in the "row" param must be natural and greater than 0.

remove(*item*)

Remove a Reply Markup item from keyboards.

Parameters
item (*bale.ReplyMarkupItem*) – The reply markup item.

remove_row(*row*)

Remove a row along with the menu keyboards located in that row.

Parameters
row (int) – The row.

> **Warning:** Your numbers in the "row" param must be natural and greater than 0.

Reply Markup Item

class bale.**ReplyMarkupItem**(*item*, *row=1*)

Bases: object

property item

The reply markup item.

Type
Union[*InlineKeyboardButton*, *MenuKeyboardButton*]

property row

The row of item.

Type
Optional[int]

Inline Keyboard Button

class bale.**InlineKeyboardButton**(*text*, *, *callback_data=None*, *url=None*, *switch_inline_query=None*, *switch_inline_query_current_chat=None*)

Bases: object

This object shows an inline keyboard button (within the message).

text

Label text on the button.

Type
str

callback_data

If set, pressing the button will prompt the user to select one of their chats, open that chat and insert the bot's username and the specified inline query in the input field. Can be empty, in which case just the bot's username will be inserted. Defaults to None.

Type
Optional[str]

url

HTTP url to be opened when the button is pressed. Defaults to None.

Type
Optional[str]

switch_inline_query

If set, pressing the button will prompt the user to select one of their chats, open that chat and insert the bot's username and the specified inline query in the input field. Can be empty, in which case just the bot's username will be inserted. Defaults to None.

Type
Optional[str]

switch_inline_query_current_chat

If set, pressing the button will insert the bot's username and the specified inline query in the current chat's input field. Can be empty, in which case only the bot's username will be inserted. Defaults to None.

Type
Optional[str]

Menu Keyboard Button

class bale.MenuKeyboardButton(*text*, *, *request_contact=False*, *request_location=False*)

Bases: object

This object shows a Keyboard Button

text

Keyboard Text.

Type
str

request_contact

If True, the user's phone number will be sent as a contact when the button is pressed.

Type
Optional[bool]

request_location

If True, the user's current location will be sent when the button is pressed. Available in private chats only.

Type
Optional[bool]

Payments

Invoice

class bale.Invoice(*title*, *description*, *start_parameter*, *currency*, *total_amount*)

Bases: BaleObject

This object shows Invoice

title

Product name.

Type
str

description

Product description.

Type
str

start_parameter

Unique bot deep-linking parameter that can be used to generate this invoice.

Type
str

currency

Three-letter ISO 4217 currency code.

Type
str

total_amount

Total price in the smallest units of the currency (integer, not float/double).

Type
int

Labeled Price

class bale.**LabeledPrice**(*label=None*, *amount=None*)

Bases: BaleObject

This object shows a LabeledPrice.

label

Label Price.

Type
Optional[str]

amount

Amount Price.

Type
Optional[int]

Successful Payment

class bale.**SuccessfulPayment**(*currency*, *total_amount*, *invoice_payload=None*, *shipping_option_id=None*)

Bases: BaleObject

This object contains basic information about a successful payment.

currency

The currency in which the transaction was made.

Type
str

total_amount

The total sum of the transaction amount.

Type
int

invoice_payload

Bot specified invoice payload.

Type
Optional[str]

shipping_option_id

Identifier of the shipping option chosen by the user.

Type
Optional[str]

property payload

Aliases for *invoice_payload*

Attachments

Audio

class bale.**Audio**(*file_id*, *file_unique_id*, *duration*, *file_name*, *title*, *mime_type*, *file_size*)

Bases: *BaseFile*

This object shows an Audio.

file_id

Identifier for this file, which can be used to download or reuse the file.

Type
str

file_unique_id

Unique identifier for this file, which is supposed to be the same over time and for different bots. Can't be used to download or reuse the file.

Type
str

duration

Duration of the audio in seconds as defined by sender.

Type
int

title

Title of the audio as defined by sender or by audio tags.

Type
Optional[str]

file_name

Original audio filename as defined by sender.

Type
Optional[str]

mime_type

MIME type of file as defined by sender.

Type
Optional[str]

file_size

File size in bytes, if known.

Type
Optional[int]

Contact

class bale.**Contact**(*phone_number*, *first_name*, *last_name*, *user_id*)

Bases: BaleObject

This object shows a Contact.

phone_number

Contact's phone number.

Type
int

first_name

Contact's first name.

Type
str

last_name

Contact's last name.

Type
Optional[str]

user_id

Contact's user identifier in Bale.

Type
Optional[int]

Document

class bale.**Document**(*file_id*, *file_unique_id*, *file_name*, *thumbnail*, *mime_type*, *file_size*)

Bases: *BaseFile*

This object shows a Document.

file_id

Identifier for this file, which can be used to download or reuse the file.

Type
str

file_unique_id

Unique identifier for this file, which is supposed to be the same over time and for different bots. Can't be used to download or reuse the file.

Type
str

thumbnail

document thumbnail as defined by sender.

Type
Optional[*bale.PhotoSize*]

file_name

Original document filename as defined by sender.

Type
Optional[str]

mime_type

MIME type of file as defined by sender.

Type
Optional[str]

file_size

File size in bytes, if known.

Type
Optional[int]

Base File

class bale.**BaseFile**(*file_id*, *file_unique_id*, *file_size*, ***kwargs*)

Bases: BaleObject

This object shows a Base File Class.

file_id

Identifier for this file, which can be used to download or reuse the file.

Type
str

file_unique_id

Unique identifier for this file, which is supposed to be the same over time and for different bots. Can't be used to download or reuse the file.

Type
str

file_size

File size in bytes, if known.

Type
Optional[int]

Note: You can get more information from the file with BaseFile.kwargs_data.

---

async get()

For the documentation of the arguments, please see *bale.Bot.get_file()*.

async save_to_memory(*out*)

Download this file into memory. out needs to be supplied with a io.BufferedIOBase, the file contents will be saved to that object using the io.BufferedIOBase.write() method.

Parameters
out (io.BinaryIO) – A file-like object. Must be opened for writing in binary mode.

to_input_file()

Converts the file to a standard object for sending/uploading it. This object is require in the file sending methods.

Returns
The *bale.InputFile* Object for send.

Return type
bale.InputFile

Input File

class bale.**InputFile**(*file*, *, *file_name=None*)

Bases: object

This object shows a file ready to send/upload.

> **Warning:** Just for upload file, you can use "file_name" param.

Examples

Attachment Bot

```
# upload the file
with open('./my_file.png', 'rb') as f:
    file = InputFile(f.read())

# use the unique file id
file = InputFile("YOUR_FILE_ID")
```

Parameters

- **file** (io.BufferedReader | str | bytes) – Your File. Pass a file_id as String to send a file that exists on the Bale servers (recommended), pass an HTTP URL as a String for Bale to get a file from the Internet, or upload a new one.
- **file_name** (Optional[str]) – Additional interface options. It is used only when uploading a file.

Location

class bale.**Location**(*longitude*, *latitude*, *horizontal_accuracy=None*)

Bases: BaleObject

This object shows a Location

longitude

Location longitude

Type
int

latitude

Location latitude

Type
int

horizontal_accuracy

The radius of uncertainty for the location, measured in meters; 0-1500.

Type
Optional[int]

property link

Export location link from Google map

Type
str

Photo Size

class bale.**PhotoSize**(*file_id*, *file_unique_id*, *width*, *height*, *file_size*)

Bases: *BaseFile*

This object represents one size of a photo or a file/sticker thumbnail.

file_id

Identifier for this photo file, which can be used to download or reuse the file.

Type
str

file_unique_id

Unique file identifier of thumbnail file.

Type
str

width

photo width.

Type
int

height

photo height.

Type
str

file_size

photo file size in bytes.

Type
Optional[int]

Video

class bale.**Video**(*file_id*, *file_unique_id*, *width*, *height*, *duration*, *file_name*, *mime_type*, *file_size*)

Bases: *BaseFile*

This object shows a Video.

file_id

Identifier for this file, which can be used to download or reuse the file.

Type
str

file_unique_id

Unique identifier for this file, which is supposed to be the same over time and for different bots. Can't be used to download or reuse the file.

Type
str

width

Video width as defined by sender.

Type
int

height

Video height as defined by sender.

Type
str

duration

Duration of the video in seconds as defined by sender.

Type
int

file_name

Original video filename as defined by sender.

Type
Optional[str]

mime_type

MIME type of file as defined by sender.

Type
Optional[str]

file_size

File size in bytes, if known.

Type
Optional[int]

Animation

class bale.**Animation**(*file_id*, *file_unique_id*, *width*, *height*, *duration*, *file_name*, *thumbnail*, *mime_type*, *file_size*)

Bases: *BaseFile*

This object shows an Animation.

file_id

Identifier for this file, which can be used to download or reuse the file.

Type
str

file_unique_id

Unique identifier for this file, which is supposed to be the same over time and for different bots. Can't be used to download or reuse the file.

Type
str

width

Animation width as defined by sender.

Type
int

height

Animation height as defined by sender.

Type
str

duration

Duration of the animation in seconds as defined by sender.

Type
int

thumbnail

Animation thumbnail as defined by sender.

Type
Optional[*bale.PhotoSize*]

file_name

Original animation filename as defined by sender.

Type
Optional[str]

mime_type

MIME type of file as defined by sender.

Type
Optional[str]

file_size

File size in bytes, if known.

Type
Optional[int]

5.2 Helpers

5.2.1 Deep Link

bale.helpers.**create_deep_linked_url**(*bot_username*, *payload*)

Creating a deep link for the bot.

> **Warning:** The username of the robot must be entered in the correct format and invalid characters should not be used in the payload parameter.

Parameters

- **bot_username** (str) – The username of bot.
- **payload** (str) – The Payload of deep link

5.2.2 Find

bale.helpers.**find**(*predicate*, *iterable*)

A helper to return the first element in the sequence that meets the predicate.

Parameters

- **predicate** – A function to return boolean-like result.
- **iterable** (str) – An iterable to search through.

5.3 Event Reference

The list of events that can be received by the bot.

An example of how to listen to events in different situations (for on_message event):

```
from bale import Bot, Message

bot = Bot("YOUR_TOKEN")

@bot.event
async def on_message(message: Message):
    if message.content == '/start':
        return await message.reply("Hi, python-bale-bot!")

bot.run()
```

```
from bale import Bot, Message

class MyBot(Bot):
    async def on_message(self, message: Message):
        if message.content == '/start':
            return await message.reply("Hi, python-bale-bot!")

MyBot('YOUR_TOKEN').run()
```

5.3.1 Connection

async on_before_ready()

This event is called before the updater starts.

async on_ready()

When the updater starts working and the Bot information is placed in *bale.Bot.user*.

5.3.2 Updates

async on_update(*update*)

This event is called when an update is received from "Bale" servers.

Parameters
update (*bale.Update*) – update received.

5.3.3 Messages

async on_message(*message*)

This event is called when sending a message in a chat to which the bot is connected.

Parameters
message (*bale.Message*) – message sent.

async on_message_edit(*message*)

This event is called when the sent message is edited.

Parameters
message (*bale.Message*) – message edited.

5.3.4 CallbackQuery

async on_callback(*callback*)

This event is called when a callback query is created.

Parameters
callback (*bale.CallbackQuery*) – callback received.

5.3.5 Groups

async on_member_chat_join(*message*, *chat*, *user*)

This event is called when a user joins the chat.

Parameters

- **message** (*bale.Message*) – message sent.
- **chat** (*bale.Chat*) – the chat.
- **user** (*bale.User*) – the user.

async on_member_chat_leave(*message*, *chat*, *user*)

This event is called when a user leaves the chat.

Parameters

- **message** (*bale.Message*) – message sent.
- **chat** (*bale.Chat*) – the chat.
- **user** (*bale.User*) – the user.

5.3.6 Payments

async on_successful_payment(*payment*)

This event is called when a transaction is completed and its status is successful.

Parameters

successful_payment (*bale.SuccessfulPayment*) – the payment.

5.4 Examples

In this section, there are some robots that are written with `python-bale-bot`.

5.4.1 examples.basic

This robot will answer you with only some commands.

5.4.2 examples.inlinemarkup

This example sheds some light on inline keyboards, callback queries and message editing.

5.4.3 examples.attachment

A basic example of a bot that can send media

5.4.4 examples.invoice

A basic example of a bot that can accept payments.

5.4.5 examples.conversation

A common task for a bot is to ask information from the user.

INDEX

N

O

P

R

S

T

U

V

W