pyTelegramBotAPI Documentation

Выпуск 4.8.0

coder2020official

нояб. 30, 2022

Оглавление

Глава 1

TeleBot

TeleBot is synchronous and asynchronous implementation of Telegram Bot API.

1.1 Chats

English chat: Private chat

Russian chat: @pytelegrambotapi_talks_ru

News: @pyTelegramBotAPI

Pypi: Pypi

Source: Github repository

1.2 Some features:

Easy to learn and use.

Easy to understand.

Both sync and async.

Examples on features.

States

And more...

1.3 Content

1.3.1 Installation Guide

Using PIP

```
$ pip install pyTelegramBotAPI
```

Using pipenv

```
$ pipenv install pyTelegramBotAPI
```

By cloning repository

```
$ git clone https://github.com/eternnoir/pyTelegramBotAPI.git
$ cd pyTelegramBotAPI
$ python setup.py install
```

Directly using pip

```
$ pip install git+https://github.com/eternnoir/pyTelegramBotAPI.git
```

It is generally recommended to use the first option.

While the API is production-ready, it is still under development and it has regular updates, do not forget to update it regularly by calling:

```
$ pip install pytelegrambotapi --upgrade
```

1.3.2 Quick start

Synchronous TeleBot

```
#!/usr/bin/python

# This is a simple echo bot using the decorator mechanism.
# It echoes any incoming text messages.

import telebot

API_TOKEN = '<api_token>'

bot = telebot.TeleBot(API_TOKEN)


# Handle '/start' and '/help'
```

(continues on next page)

(продолжение с предыдущей страницы)

```
@bot.message_handler(commands=['help', 'start'])
def send_welcome(message):
    bot.reply_to(message, """\
Hi there, I am EchoBot.
I am here to echo your kind words back to you. Just say anything nice and I'll say the␣
↪exact same thing to you!\
""")


# Handle all other messages with content_type 'text' (content_types defaults to ['text'])
@bot.message_handler(func=lambda message: True)
def echo_message(message):
    bot.reply_to(message, message.text)


bot.infinity_polling()
```

Asynchronous TeleBot

```
#!/usr/bin/python

# This is a simple echo bot using the decorator mechanism.
# It echoes any incoming text messages.

from telebot.async_telebot import AsyncTeleBot
bot = AsyncTeleBot('TOKEN')



# Handle '/start' and '/help'
@bot.message_handler(commands=['help', 'start'])
async def send_welcome(message):
    await bot.reply_to(message, """\
Hi there, I am EchoBot.
I am here to echo your kind words back to you. Just say anything nice and I'll say the␣
↪exact same thing to you!\
""")


# Handle all other messages with content_type 'text' (content_types defaults to ['text'])
@bot.message_handler(func=lambda message: True)
async def echo_message(message):
    await bot.reply_to(message, message.text)


import asyncio
asyncio.run(bot.polling())
```

1.3.3 Types of API

class telebot.types.Animation(*file_id*, *file_unique_id*, *width=None*, *height=None*, *duration=None*, *thumb=None*, *file_name=None*, *mime_type=None*, *file_size=None*, ***kwargs*)

Базовые классы: *JsonDeserializable*

This object represents an animation file (GIF or H.264/MPEG-4 AVC video without sound).

Telegram Documentation: https://core.telegram.org/bots/api#animation

Параметры

- `file_id` (`str`) – Identifier for this file, which can be used to download or reuse the file
- `file_unique_id` (`str`) – 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.
- `width` (`int`) – Video width as defined by sender
- `height` (`int`) – Video height as defined by sender
- `duration` (`int`) – Duration of the video in seconds as defined by sender
- `thumb` (*telebot.types.PhotoSize*) – Optional. Animation thumbnail as defined by sender
- `file_name` (`str`) – Optional. Original animation filename as defined by sender
- `mime_type` (`str`) – Optional. MIME type of the file as defined by sender
- `file_size` (`int`) – Optional. File size in bytes. It can be bigger than 2^31 and some programming languages may have difficulty/silent defects in interpreting it. But it has at most 52 significant bits, so a signed 64-bit integer or double-precision float type are safe for storing this value.

Результат
Instance of the class

Тип результата
telebot.types.Animation

class telebot.types.Audio(*file_id*, *file_unique_id*, *duration*, *performer=None*, *title=None*, *file_name=None*, *mime_type=None*, *file_size=None*, *thumb=None*, ***kwargs*)

Базовые классы: *JsonDeserializable*

This object represents an audio file to be treated as music by the Telegram clients.

Telegram Documentation: https://core.telegram.org/bots/api#audio

Параметры

- `file_id` (`str`) – Identifier for this file, which can be used to download or reuse the file
- `file_unique_id` (`str`) – 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.
- `duration` (`int`) – Duration of the audio in seconds as defined by sender
- `performer` (`str`) – Optional. Performer of the audio as defined by sender or by audio tags

- `title` (`str`) – Optional. Title of the audio as defined by sender or by audio tags
- `file_name` (`str`) – Optional. Original filename as defined by sender
- `mime_type` (`str`) – Optional. MIME type of the file as defined by sender
- `file_size` (`int`) – Optional. File size in bytes. It can be bigger than 2^31 and some programming languages may have difficulty/silent defects in interpreting it. But it has at most 52 significant bits, so a signed 64-bit integer or double-precision float type are safe for storing this value.
- `thumb` (*telebot.types.PhotoSize*) – Optional. Thumbnail of the album cover to which the music file belongs

Результат
Instance of the class

Тип результата
telebot.types.Audio

`class telebot.types.BotCommand(`*command*, *description*`)`

Базовые классы: *JsonSerializable*, *JsonDeserializable*, *Dictionaryable*

This object represents a bot command.

Telegram Documentation: https://core.telegram.org/bots/api#botcommand

Параметры

- `command` (`str`) – Text of the command; 1-32 characters. Can contain only lowercase English letters, digits and underscores.
- `description` (`str`) – Description of the command; 1-256 characters.

Результат
Instance of the class

Тип результата
telebot.types.BotCommand

`class telebot.types.BotCommandScope(`*type='default'*, *chat_id=None*, *user_id=None*`)`

Базовые классы: `ABC`, *JsonSerializable*

This object represents the scope to which bot commands are applied. Currently, the following 7 scopes are supported:

- *BotCommandScopeDefault*
- *BotCommandScopeAllPrivateChats*
- *BotCommandScopeAllGroupChats*
- *BotCommandScopeAllChatAdministrators*
- *BotCommandScopeChat*
- *BotCommandScopeChatAdministrators*
- *BotCommandScopeChatMember*

Determining list of commands The following algorithm is used to determine the list of commands for a particular user viewing the bot menu. The first list of commands which is set is returned:

Commands in the chat with the bot:

- *BotCommandScopeChat* + language_code

- *BotCommandScopeChat*
- *BotCommandScopeAllPrivateChats* + language_code
- *BotCommandScopeAllPrivateChats*
- *BotCommandScopeDefault* + language_code
- *BotCommandScopeDefault*

Commands in group and supergroup chats:

- *BotCommandScopeChatMember* + language_code
- *BotCommandScopeChatMember*
- *BotCommandScopeChatAdministrators* + language_code (administrators only)
- *BotCommandScopeChatAdministrators* (administrators only)
- *BotCommandScopeChat* + language_code
- *BotCommandScopeChat*
- *BotCommandScopeAllChatAdministrators* + language_code (administrators only)
- *BotCommandScopeAllChatAdministrators* (administrators only)
- *BotCommandScopeAllGroupChats* + language_code
- *BotCommandScopeAllGroupChats*
- *BotCommandScopeDefault* + language_code
- *BotCommandScopeDefault*

Результат
Instance of the class

Тип результата
telebot.types.BotCommandScope

class telebot.types.BotCommandScopeAllChatAdministrators

Базовые классы: *BotCommandScope*

Represents the scope of bot commands, covering all group and supergroup chat administrators.

Telegram Documentation: https://core.telegram.org/bots/api#botcommandscopeallchatadministrators

Параметры
`type` (`str`) – Scope type, must be all_chat_administrators

Результат
Instance of the class

Тип результата
telebot.types.BotCommandScopeAllChatAdministrators

class telebot.types.BotCommandScopeAllGroupChats

Базовые классы: *BotCommandScope*

Represents the scope of bot commands, covering all group and supergroup chats.

Telegram Documentation: https://core.telegram.org/bots/api#botcommandscopeallgroupchats

Параметры
`type` (`str`) – Scope type, must be all_group_chats

Результат
Instance of the class

Тип результата
telebot.types.BotCommandScopeAllGroupChats

class telebot.types.BotCommandScopeAllPrivateChats

Базовые классы: *BotCommandScope*

Represents the scope of bot commands, covering all private chats.

Telegram Documentation: https://core.telegram.org/bots/api#botcommandscopeallprivatechats

Параметры
type (str) – Scope type, must be all_private_chats

Результат
Instance of the class

Тип результата
telebot.types.BotCommandScopeAllPrivateChats

class telebot.types.BotCommandScopeChat(*chat_id=None*)

Базовые классы: *BotCommandScope*

Represents the scope of bot commands, covering a specific chat.

Telegram Documentation: https://core.telegram.org/bots/api#botcommandscopechat

Параметры

- type (str) – Scope type, must be chat
- chat_id (int or str) – Unique identifier for the target chat or username of the target supergroup (in the format @supergroupusername)

Результат
Instance of the class

Тип результата
telebot.types.BotCommandScopeChat

class telebot.types.BotCommandScopeChatAdministrators(*chat_id=None*)

Базовые классы: *BotCommandScope*

Represents the scope of bot commands, covering all administrators of a specific group or supergroup chat.

Telegram Documentation: https://core.telegram.org/bots/api#botcommandscopechatadministrators

Параметры

- type (str) – Scope type, must be chat_administrators
- chat_id (int or str) – Unique identifier for the target chat or username of the target supergroup (in the format @supergroupusername)

Результат
Instance of the class

Тип результата
telebot.types.BotCommandScopeChatAdministrators

class telebot.types.BotCommandScopeChatMember(*chat_id=None*, *user_id=None*)

Базовые классы: *BotCommandScope*

Represents the scope of bot commands, covering a specific member of a group or supergroup chat.

Telegram Documentation: https://core.telegram.org/bots/api#botcommandscopechatmember

Параметры

- `type` (`str`) – Scope type, must be chat_member
- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target supergroup (in the format @supergroupusername)
- `user_id` (`int`) – Unique identifier of the target user

Результат
Instance of the class

Тип результата
telebot.types.BotCommandScopeChatMember

class telebot.types.BotCommandScopeDefault

Базовые классы: *BotCommandScope*

Represents the default scope of bot commands. Default commands are used if no commands with a narrower scope are specified for the user.

Telegram Documentation: https://core.telegram.org/bots/api#botcommandscopedefault

Параметры
`type` (`str`) – Scope type, must be default

Результат
Instance of the class

Тип результата
telebot.types.BotCommandScopeDefault

class telebot.types.CallbackQuery(*id*, *from_user*, *data*, *chat_instance*, *json_string*, *message=None*, *inline_message_id=None*, *game_short_name=None*, ***kwargs*)

Базовые классы: *JsonDeserializable*

This object represents an incoming callback query from a callback button in an inline keyboard. If the button that originated the query was attached to a message sent by the bot, the field message will be present. If the button was attached to a message sent via the bot (in inline mode), the field inline_message_id will be present. Exactly one of the fields data or game_short_name will be present.

Telegram Documentation: https://core.telegram.org/bots/api#callbackquery

Параметры

- `id` (`str`) – Unique identifier for this query
- `from_user` (*telebot.types.User*) – Sender
- `message` (*telebot.types.Message*) – Optional. 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
- `inline_message_id` (`str`) – Optional. Identifier of the message sent via the bot in inline mode, that originated the query.

- `chat_instance` (`str`) – Global identifier, uniquely corresponding to the chat to which the message with the callback button was sent. Useful for high scores in games.
- `data` (`str`) – Optional. Data associated with the callback button. Be aware that the message originated the query can contain no callback buttons with this data.
- `game_short_name` (`str`) – Optional. Short name of a Game to be returned, serves as the unique identifier for the game

Результат
Instance of the class

Тип результата
telebot.types.CallbackQuery

class telebot.types.Chat(*id*, *type*, *title=None*, *username=None*, *first_name=None*, *last_name=None*, *photo=None*, *bio=None*, *has_private_forwards=None*, *description=None*, *invite_link=None*, *pinned_message=None*, *permissions=None*, *slow_mode_delay=None*, *message_auto_delete_time=None*, *has_protected_content=None*, *sticker_set_name=None*, *can_set_sticker_set=None*, *linked_chat_id=None*, *location=None*, *join_to_send_messages=None*, *join_by_request=None*, *has_restricted_voice_and_video_messages=None*, *is_forum=None*, *active_usernames=None*, *emoji_status_custom_emoji_id=None*, ***kwargs*)

Базовые классы: *JsonDeserializable*

This object represents a chat.

Telegram Documentation: https://core.telegram.org/bots/api#chat

Параметры

- `id` (`int`) – Unique identifier for this chat. This number may have more than 32 significant bits and some programming languages may have difficulty/silent defects in interpreting it. But it has at most 52 significant bits, so a signed 64-bit integer or double-precision float type are safe for storing this identifier.
- `type` (`str`) – Type of chat, can be either “private”, “group”, “supergroup” or “channel”
- `title` (`str`) – Optional. Title, for supergroups, channels and group chats
- `username` (`str`) – Optional. Username, for private chats, supergroups and channels if available
- `first_name` (`str`) – Optional. First name of the other party in a private chat
- `last_name` (`str`) – Optional. Last name of the other party in a private chat
- `is_forum` (`bool`) – Optional. True, if the supergroup chat is a forum (has topics enabled)
- `photo` (*telebot.types.ChatPhoto*) – Optional. Chat photo. Returned only in getChat.
- `active_usernames` (`list` of `str`) – Optional. If non-empty, the list of all active chat usernames; for private chats, supergroups and channels. Returned only in getChat.
- `emoji_status_custom_emoji_id` (`str`) – Optional. Custom emoji identifier of emoji status of the other party in a private chat. Returned only in getChat.

- `bio` (`str`) – Optional. Bio of the other party in a private chat. Returned only in getChat.
- `has_private_forwards` (`bool`) – Optional. `bool`, if privacy settings of the other party in the private chat allows to use tg://user?id=<user_id> links only in chats with the user. Returned only in getChat.
- `has_restricted_voice_and_video_messages` – Optional. True, if the privacy settings of the other party restrict sending voice and video note messages in the private chat. Returned only in getChat.

:type `bool`

Параметры

- `join_to_send_messages` (`bool`) – Optional. `bool`, if users need to join the supergroup before they can send messages. Returned only in getChat.
- `join_by_request` (`bool`) – Optional. `bool`, if all users directly joining the supergroup need to be approved by supergroup administrators. Returned only in getChat.
- `description` (`str`) – Optional. Description, for groups, supergroups and channel chats. Returned only in getChat.
- `invite_link` (`str`) – Optional. Primary invite link, for groups, supergroups and channel chats. Returned only in getChat.
- `pinned_message` (*telebot.types.Message*) – Optional. The most recent pinned message (by sending date). Returned only in getChat.
- `permissions` (*telebot.types.ChatPermissions*) – Optional. Default chat member permissions, for groups and supergroups. Returned only in getChat.
- `slow_mode_delay` (`int`) – Optional. For supergroups, the minimum allowed delay between consecutive messages sent by each unpriviledged user; in seconds. Returned only in getChat.
- `message_auto_delete_time` (`int`) – Optional. The time after which all messages sent to the chat will be automatically deleted; in seconds. Returned only in getChat.
- `has_protected_content` (`bool`) – Optional. `bool`, if messages from the chat can't be forwarded to other chats. Returned only in getChat.
- `sticker_set_name` (`str`) – Optional. For supergroups, name of group sticker set. Returned only in getChat.
- `can_set_sticker_set` (`bool`) – Optional. `bool`, if the bot can change the group sticker set. Returned only in getChat.
- `linked_chat_id` (`int`) – Optional. Unique identifier for the linked chat, i.e. the discussion group identifier for a channel and vice versa; for supergroups and channel chats. This identifier may be greater than 32 bits and some programming languages may have difficulty/silent defects in interpreting it. But it is smaller than 52 bits, so a signed 64 bit integer or double-precision float type are safe for storing this identifier. Returned only in getChat.
- `location` (*telebot.types.ChatLocation*) – Optional. For supergroups, the location to which the supergroup is connected. Returned only in getChat.

Результат

Instance of the class

Тип результата
telebot.types.Chat

class telebot.types.ChatAdministratorRights(*is_anonymous: bool*, *can_manage_chat: bool*, *can_delete_messages: bool*, *can_manage_video_chats: bool*, *can_restrict_members: bool*, *can_promote_members: bool*, *can_change_info: bool*, *can_invite_users: bool*, *can_post_messages: Optional[bool] = None*, *can_edit_messages: Optional[bool] = None*, *can_pin_messages: Optional[bool] = None*, *can_manage_topics: Optional[bool] = None*)

Базовые классы: *JsonDeserializable*, *JsonSerializable*, *Dictionaryable*

Represents the rights of an administrator in a chat.

Telegram Documentation: https://core.telegram.org/bots/api#chatadministratorrights

Параметры

- `is_anonymous` (`bool`) – True, if the user's presence in the chat is hidden
- `can_manage_chat` (`bool`) – True, if the administrator can access the chat event log, chat statistics, message statistics in channels, see channel members, see anonymous administrators in supergroups and ignore slow mode. Implied by any other administrator privilege
- `can_delete_messages` (`bool`) – True, if the administrator can delete messages of other users
- `can_manage_video_chats` (`bool`) – True, if the administrator can manage video chats
- `can_restrict_members` (`bool`) – True, if the administrator can restrict, ban or unban chat members
- `can_promote_members` (`bool`) – True, if the administrator can add new administrators with a subset of their own privileges or demote administrators that he has promoted, directly or indirectly (promoted by administrators that were appointed by the user)
- `can_change_info` (`bool`) – True, if the user is allowed to change the chat title, photo and other settings
- `can_invite_users` (`bool`) – True, if the user is allowed to invite new users to the chat
- `can_post_messages` (`bool`) – Optional. True, if the administrator can post in the channel; channels only
- `can_edit_messages` (`bool`) – Optional. True, if the administrator can edit messages of other users and can pin messages; channels only
- `can_pin_messages` (`bool`) – Optional. True, if the user is allowed to pin messages; groups and supergroups only
- `can_manage_topics` (`bool`) – Optional. True, if the user is allowed to create, rename, close, and reopen forum topics; supergroups only

Результат
Instance of the class

Тип результата
telebot.types.ChatAdministratorRights

class telebot.types.ChatInviteLink(*invite_link*, *creator*, *creates_join_request*, *is_primary*, *is_revoked*, *name=None*, *expire_date=None*, *member_limit=None*, *pending_join_request_count=None*, ***kwargs*)

Базовые классы: *JsonSerializable*, *JsonDeserializable*, *Dictionaryable*

Represents an invite link for a chat.

Telegram Documentation: https://core.telegram.org/bots/api#chatinvitelink

Параметры

- `invite_link` (`str`) – The invite link. If the link was created by another chat administrator, then the second part of the link will be replaced with "...".
- `creator` (*telebot.types.User*) – Creator of the link
- `creates_join_request` (`bool`) – True, if users joining the chat via the link need to be approved by chat administrators
- `is_primary` (`bool`) – True, if the link is primary
- `is_revoked` (`bool`) – True, if the link is revoked
- `name` (`str`) – Optional. Invite link name
- `expire_date` (`int`) – Optional. Point in time (Unix timestamp) when the link will expire or has been expired
- `member_limit` (`int`) – Optional. The maximum number of users that can be members of the chat simultaneously after joining the chat via this invite link; 1-99999
- `pending_join_request_count` (`int`) – Optional. Number of pending join requests created using this link

Результат
Instance of the class

Тип результата
telebot.types.ChatInviteLink

class telebot.types.ChatJoinRequest(*chat*, *from_user*, *date*, *bio=None*, *invite_link=None*, ***kwargs*)

Базовые классы: *JsonDeserializable*

Represents a join request sent to a chat.

Telegram Documentation: https://core.telegram.org/bots/api#chatjoinrequest

Параметры

- `chat` (*telebot.types.Chat*) – Chat to which the request was sent
- `from` – User that sent the join request
- `date` (`int`) – Date the request was sent in Unix time
- `bio` (`str`) – Optional. Bio of the user.
- `invite_link` (*telebot.types.ChatInviteLink*) – Optional. Chat invite link that was used by the user to send the join request

Результат
Instance of the class

Тип результата
telebot.types.ChatJoinRequest

class telebot.types.ChatLocation(*location*, *address*, ***kwargs*)

Базовые классы: *JsonSerializable*, *JsonDeserializable*, *Dictionaryable*

Represents a location to which a chat is connected.

Telegram Documentation: https://core.telegram.org/bots/api#chatlocation

Параметры

- `location` (*telebot.types.Location*) – The location to which the supergroup is connected. Can't be a live location.
- `address` (`str`) – Location address; 1-64 characters, as defined by the chat owner

Результат
Instance of the class

Тип результата
telebot.types.ChatLocation

class telebot.types.ChatMember(*user*, *status*, *custom_title=None*, *is_anonymous=None*, *can_be_edited=None*, *can_post_messages=None*, *can_edit_messages=None*, *can_delete_messages=None*, *can_restrict_members=None*, *can_promote_members=None*, *can_change_info=None*, *can_invite_users=None*, *can_pin_messages=None*, *is_member=None*, *can_send_messages=None*, *can_send_media_messages=None*, *can_send_polls=None*, *can_send_other_messages=None*, *can_add_web_page_previews=None*, *can_manage_chat=None*, *can_manage_video_chats=None*, *until_date=None*, *can_manage_topics=None*, ***kwargs*)

Базовые классы: *JsonDeserializable*

This object contains information about one member of a chat. Currently, the following 6 types of chat members are supported:

- *telebot.types.ChatMemberOwner*
- *telebot.types.ChatMemberAdministrator*
- *telebot.types.ChatMemberMember*
- *telebot.types.ChatMemberRestricted*
- *telebot.types.ChatMemberLeft*
- *telebot.types.ChatMemberBanned*

Telegram Documentation: https://core.telegram.org/bots/api#chatmember

class telebot.types.ChatMemberAdministrator(*user*, *status*, *custom_title=None*, *is_anonymous=None*, *can_be_edited=None*, *can_post_messages=None*, *can_edit_messages=None*, *can_delete_messages=None*, *can_restrict_members=None*, *can_promote_members=None*, *can_change_info=None*, *can_invite_users=None*, *can_pin_messages=None*, *is_member=None*, *can_send_messages=None*, *can_send_media_messages=None*, *can_send_polls=None*, *can_send_other_messages=None*, *can_add_web_page_previews=None*, *can_manage_chat=None*, *can_manage_video_chats=None*, *until_date=None*, *can_manage_topics=None*, ***kwargs*)

Базовые классы: *ChatMember*

Represents a chat member that has some additional privileges.

Telegram Documentation: https://core.telegram.org/bots/api#chatmemberadministrator

Параметры

- `status` (`str`) – The member's status in the chat, always "administrator"
- `user` (*telebot.types.User*) – Information about the user
- `can_be_edited` (`bool`) – True, if the bot is allowed to edit administrator privileges of that user
- `is_anonymous` (`bool`) – True, if the user's presence in the chat is hidden
- `can_manage_chat` (`bool`) – True, if the administrator can access the chat event log, chat statistics, message statistics in channels, see channel members, see anonymous administrators in supergroups and ignore slow mode. Implied by any other administrator privilege
- `can_delete_messages` (`bool`) – True, if the administrator can delete messages of other users
- `can_manage_video_chats` (`bool`) – True, if the administrator can manage video chats
- `can_restrict_members` (`bool`) – True, if the administrator can restrict, ban or unban chat members
- `can_promote_members` (`bool`) – True, if the administrator can add new administrators with a subset of their own privileges or demote administrators that he has promoted, directly or indirectly (promoted by administrators that were appointed by the user)
- `can_change_info` (`bool`) – True, if the user is allowed to change the chat title, photo and other settings
- `can_invite_users` (`bool`) – True, if the user is allowed to invite new users to the chat
- `can_post_messages` (`bool`) – Optional. True, if the administrator can post in the channel; channels only

- `can_edit_messages` (`bool`) – Optional. True, if the administrator can edit messages of other users and can pin messages; channels only
- `can_pin_messages` (`bool`) – Optional. True, if the user is allowed to pin messages; groups and supergroups only
- `can_manage_topics` (`bool`) – Optional. True, if the user is allowed to create, rename, close, and reopen forum topics; supergroups only
- `custom_title` (`str`) – Optional. Custom title for this user

Результат
Instance of the class

Тип результата
telebot.types.ChatMemberAdministrator

class telebot.types.ChatMemberBanned(*user*, *status*, *custom_title=None*, *is_anonymous=None*, *can_be_edited=None*, *can_post_messages=None*, *can_edit_messages=None*, *can_delete_messages=None*, *can_restrict_members=None*, *can_promote_members=None*, *can_change_info=None*, *can_invite_users=None*, *can_pin_messages=None*, *is_member=None*, *can_send_messages=None*, *can_send_media_messages=None*, *can_send_polls=None*, *can_send_other_messages=None*, *can_add_web_page_previews=None*, *can_manage_chat=None*, *can_manage_video_chats=None*, *until_date=None*, *can_manage_topics=None*, ***kwargs*)

Базовые классы: *ChatMember*

Represents a chat member that was banned in the chat and can't return to the chat or view chat messages.

Telegram Documentation: https://core.telegram.org/bots/api#chatmemberbanned

Параметры

- `status` (`str`) – The member's status in the chat, always "kicked"
- `user` (*telebot.types.User*) – Information about the user
- `until_date` (`int`) – Date when restrictions will be lifted for this user; unix time. If 0, then the user is banned forever

Результат
Instance of the class

Тип результата
telebot.types.ChatMemberBanned

class telebot.types.ChatMemberLeft(*user*, *status*, *custom_title=None*, *is_anonymous=None*, *can_be_edited=None*, *can_post_messages=None*, *can_edit_messages=None*, *can_delete_messages=None*, *can_restrict_members=None*, *can_promote_members=None*, *can_change_info=None*, *can_invite_users=None*, *can_pin_messages=None*, *is_member=None*, *can_send_messages=None*, *can_send_media_messages=None*, *can_send_polls=None*, *can_send_other_messages=None*, *can_add_web_page_previews=None*, *can_manage_chat=None*, *can_manage_video_chats=None*, *until_date=None*, *can_manage_topics=None*, ***kwargs*)

Базовые классы: *ChatMember*

Represents a chat member that isn't currently a member of the chat, but may join it themselves.

Telegram Documentation: https://core.telegram.org/bots/api#chatmemberleft

Параметры

- `status` (`str`) – The member's status in the chat, always "left"
- `user` (*telebot.types.User*) – Information about the user

Результат
Instance of the class

Тип результата
telebot.types.ChatMemberLeft

`class telebot.types.ChatMemberMember(`*user*, *status*, *custom_title=None*, *is_anonymous=None*, *can_be_edited=None*, *can_post_messages=None*, *can_edit_messages=None*, *can_delete_messages=None*, *can_restrict_members=None*, *can_promote_members=None*, *can_change_info=None*, *can_invite_users=None*, *can_pin_messages=None*, *is_member=None*, *can_send_messages=None*, *can_send_media_messages=None*, *can_send_polls=None*, *can_send_other_messages=None*, *can_add_web_page_previews=None*, *can_manage_chat=None*, *can_manage_video_chats=None*, *until_date=None*, *can_manage_topics=None*, ***kwargs*`)`

Базовые классы: *ChatMember*

Represents a chat member that has no additional privileges or restrictions.

Telegram Documentation: https://core.telegram.org/bots/api#chatmembermember

Параметры

- `status` (`str`) – The member's status in the chat, always "member"
- `user` (*telebot.types.User*) – Information about the user

Результат
Instance of the class

Тип результата
telebot.types.ChatMemberMember

`class telebot.types.ChatMemberOwner(`*user*, *status*, *custom_title=None*, *is_anonymous=None*, *can_be_edited=None*, *can_post_messages=None*, *can_edit_messages=None*, *can_delete_messages=None*, *can_restrict_members=None*, *can_promote_members=None*, *can_change_info=None*, *can_invite_users=None*, *can_pin_messages=None*, *is_member=None*, *can_send_messages=None*, *can_send_media_messages=None*, *can_send_polls=None*, *can_send_other_messages=None*, *can_add_web_page_previews=None*, *can_manage_chat=None*, *can_manage_video_chats=None*, *until_date=None*, *can_manage_topics=None*, ***kwargs*`)`

Базовые классы: *ChatMember*

Represents a chat member that owns the chat and has all administrator privileges.

Telegram Documentation: https://core.telegram.org/bots/api#chatmemberowner

Параметры

- `status` (`str`) – The member's status in the chat, always "creator"
- `user` (*telebot.types.User*) – Information about the user
- `is_anonymous` (`bool`) – True, if the user's presence in the chat is hidden
- `custom_title` (`str`) – Optional. Custom title for this user

Результат
Instance of the class

Тип результата
telebot.types.ChatMemberOwner

```
class telebot.types.ChatMemberRestricted(user, status, custom_title=None, is_anonymous=None,
                                         can_be_edited=None, can_post_messages=None,
                                         can_edit_messages=None, can_delete_messages=None,
                                         can_restrict_members=None,
                                         can_promote_members=None, can_change_info=None,
                                         can_invite_users=None, can_pin_messages=None,
                                         is_member=None, can_send_messages=None,
                                         can_send_media_messages=None,
                                         can_send_polls=None,
                                         can_send_other_messages=None,
                                         can_add_web_page_previews=None,
                                         can_manage_chat=None,
                                         can_manage_video_chats=None, until_date=None,
                                         can_manage_topics=None, **kwargs)
```

Базовые классы: *ChatMember*

Represents a chat member that is under certain restrictions in the chat. Supergroups only.

Telegram Documentation: https://core.telegram.org/bots/api#chatmemberrestricted

Параметры

- `status` (`str`) – The member's status in the chat, always "restricted"
- `user` (*telebot.types.User*) – Information about the user
- `is_member` (`bool`) – True, if the user is a member of the chat at the moment of the request
- `can_change_info` (`bool`) – True, if the user is allowed to change the chat title, photo and other settings
- `can_invite_users` (`bool`) – True, if the user is allowed to invite new users to the chat
- `can_pin_messages` (`bool`) – True, if the user is allowed to pin messages
- `can_manage_topics` (`bool`) – True, if the user is allowed to create forum topics
- `can_send_messages` (`bool`) – True, if the user is allowed to send text messages, contacts, locations and venues
- `can_send_media_messages` (`bool`) – True, if the user is allowed to send audios, documents, photos, videos, video notes and voice notes

- `can_send_polls` (`bool`) – True, if the user is allowed to send polls
- `can_send_other_messages` (`bool`) – True, if the user is allowed to send animations, games, stickers and use inline bots
- `can_add_web_page_previews` (`bool`) – True, if the user is allowed to add web page previews to their messages
- `until_date` (`int`) – Date when restrictions will be lifted for this user; unix time. If 0, then the user is restricted forever

Результат
Instance of the class

Тип результата
telebot.types.ChatMemberRestricted

class telebot.types.ChatMemberUpdated(*chat*, *from_user*, *date*, *old_chat_member*, *new_chat_member*, *invite_link=None*, ***kwargs*)

Базовые классы: *JsonDeserializable*

This object represents changes in the status of a chat member.

Telegram Documentation: https://core.telegram.org/bots/api#chatmemberupdated

Параметры

- `chat` (*telebot.types.Chat*) – Chat the user belongs to
- `from_user` (*telebot.types.User*) – Performer of the action, which resulted in the change
- `date` (`int`) – Date the change was done in Unix time
- `old_chat_member` (*telebot.types.ChatMember*) – Previous information about the chat member
- `new_chat_member` (*telebot.types.ChatMember*) – New information about the chat member
- `invite_link` (*telebot.types.ChatInviteLink*) – Optional. Chat invite link, which was used by the user to join the chat; for joining by invite link events only.

Результат
Instance of the class

Тип результата
telebot.types.ChatMemberUpdated

property difference: Dict[str, List]

Get the difference between *old_chat_member* and *new_chat_member* as a dict in the following format {„parameter“: [old_value, new_value]} E.g {„status“: [„member“, „kicked“], „until_date“: [None, 1625055092]}

Результат
Dict of differences

Тип результата
Dict[str, List]

class telebot.types.ChatPermissions(*can_send_messages=None*, *can_send_media_messages=None*, *can_send_polls=None*, *can_send_other_messages=None*, *can_add_web_page_previews=None*, *can_change_info=None*, *can_invite_users=None*, *can_pin_messages=None*, *can_manage_topics=None*, ***kwargs*)

Базовые классы: *JsonDeserializable*, *JsonSerializable*, *Dictionaryable*

Describes actions that a non-administrator user is allowed to take in a chat.

Telegram Documentation: https://core.telegram.org/bots/api#chatpermissions

Параметры

- `can_send_messages` (`bool`) – Optional. True, if the user is allowed to send text messages, contacts, locations and venues
- `can_send_media_messages` (`bool`) – Optional. True, if the user is allowed to send audios, documents, photos, videos, video notes and voice notes, implies can_send_messages
- `can_send_polls` (`bool`) – Optional. True, if the user is allowed to send polls, implies can_send_messages
- `can_send_other_messages` (`bool`) – Optional. True, if the user is allowed to send animations, games, stickers and use inline bots, implies can_send_media_messages
- `can_add_web_page_previews` (`bool`) – Optional. True, if the user is allowed to add web page previews to their messages, implies can_send_media_messages
- `can_change_info` (`bool`) – Optional. True, if the user is allowed to change the chat title, photo and other settings. Ignored in public supergroups
- `can_invite_users` (`bool`) – Optional. True, if the user is allowed to invite new users to the chat
- `can_pin_messages` (`bool`) – Optional. True, if the user is allowed to pin messages. Ignored in public supergroups
- `can_manage_topics` (`bool`) – Optional. True, if the user is allowed to create forum topics. If omitted defaults to the value of can_pin_messages

Результат

Instance of the class

Тип результата

telebot.types.ChatPermissions

`class telebot.types.ChatPhoto(`*small_file_id*, *small_file_unique_id*, *big_file_id*, *big_file_unique_id*, ***kwargs*`)`

Базовые классы: *JsonDeserializable*

This object represents a chat photo.

Telegram Documentation: https://core.telegram.org/bots/api#chatphoto

Параметры

- `small_file_id` (`str`) – File identifier of small (160x160) chat photo. This file_id can be used only for photo download and only for as long as the photo is not changed.
- `small_file_unique_id` (`str`) – Unique file identifier of small (160x160) chat photo, which is supposed to be the same over time and for different bots. Can't be used to download or reuse the file.
- `big_file_id` (`str`) – File identifier of big (640x640) chat photo. This file_id can be used only for photo download and only for as long as the photo is not changed.

- `big_file_unique_id` (`str`) – Unique file identifier of big (640x640) chat photo, which is supposed to be the same over time and for different bots. Can't be used to download or reuse the file.

Результат
Instance of the class

Тип результата
telebot.types.ChatPhoto

class telebot.types.ChosenInlineResult(*result_id*, *from_user*, *query*, *location=None*, *inline_message_id=None*, ***kwargs*)

Базовые классы: *JsonDeserializable*

Represents a result of an inline query that was chosen by the user and sent to their chat partner.

Telegram Documentation: https://core.telegram.org/bots/api#choseninlineresult

Параметры

- `result_id` (`str`) – The unique identifier for the result that was chosen
- `from` (*telebot.types.User*) – The user that chose the result
- `location` (*telebot.types.Location*) – Optional. Sender location, only for bots that require user location
- `inline_message_id` (`str`) – Optional. Identifier of the sent inline message. Available only if there is an inline keyboard attached to the message. Will be also received in callback queries and can be used to edit the message.
- `query` (`str`) – The query that was used to obtain the result

Результат
Instance of the class

Тип результата
telebot.types.ChosenInlineResult

class telebot.types.Contact(*phone_number*, *first_name*, *last_name=None*, *user_id=None*, *vcard=None*, ***kwargs*)

Базовые классы: *JsonDeserializable*

This object represents a phone contact.

Telegram Documentation: https://core.telegram.org/bots/api#contact

Параметры

- `phone_number` (`str`) – Contact's phone number
- `first_name` (`str`) – Contact's first name
- `last_name` (`str`) – Optional. Contact's last name
- `user_id` (`int`) – Optional. Contact's user identifier in Telegram. This number may have more than 32 significant bits and some programming languages may have difficulty/silent defects in interpreting it. But it has at most 52 significant bits, so a 64-bit integer or double-precision float type are safe for storing this identifier.
- `vcard` (`str`) – Optional. Additional data about the contact in the form of a vCard

Результат
Instance of the class

Тип результата
telebot.types.Contact

class telebot.types.Dice(*value*, *emoji*, ***kwargs*)

Базовые классы: *JsonSerializable*, *Dictionaryable*, *JsonDeserializable*

This object represents an animated emoji that displays a random value.

Telegram Documentation: https://core.telegram.org/bots/api#dice

Параметры

- `emoji` (`str`) – Emoji on which the dice throw animation is based
- `value` (`int`) – Value of the dice, 1-6 for “”, “” and “” base emoji, 1-5 for “” and “” base emoji, 1-64 for “” base emoji

Результат
Instance of the class

Тип результата
telebot.types.Dice

class telebot.types.Dictionaryable

Базовые классы: `object`

Subclasses of this class are guaranteed to be able to be converted to dictionary. All subclasses of this class must override to_dict.

class telebot.types.Document(*file_id*, *file_unique_id*, *thumb=None*, *file_name=None*, *mime_type=None*, *file_size=None*, ***kwargs*)

Базовые классы: *JsonDeserializable*

This object represents a general file (as opposed to photos, voice messages and audio files).

Telegram Documentation: https://core.telegram.org/bots/api#document

Параметры

- `file_id` (`str`) – Identifier for this file, which can be used to download or reuse the file
- `file_unique_id` (`str`) – 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.
- `thumb` (*telebot.types.PhotoSize*) – Optional. Document thumbnail as defined by sender
- `file_name` (`str`) – Optional. Original filename as defined by sender
- `mime_type` (`str`) – Optional. MIME type of the file as defined by sender
- `file_size` (`int`) – Optional. File size in bytes. It can be bigger than 2^31 and some programming languages may have difficulty/silent defects in interpreting it. But it has at most 52 significant bits, so a signed 64-bit integer or double-precision float type are safe for storing this value.

Результат
Instance of the class

Тип результата
telebot.types.Document

class telebot.types.File(*file_id*, *file_unique_id*, *file_size=None*, *file_path=None*, ***kwargs*)

Базовые классы: *JsonDeserializable*

This object represents a file ready to be downloaded. The file can be downloaded via the link https://api.telegram.org/file/bot<token>/<file_path>. It is guaranteed that the link will be valid for at least 1 hour. When the link expires, a new one can be requested by calling getFile.

Telegram Documentation: https://core.telegram.org/bots/api#file

Параметры

- `file_id` (`str`) – Identifier for this file, which can be used to download or reuse the file
- `file_unique_id` (`str`) – 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.
- `file_size` (`int`) – Optional. File size in bytes. It can be bigger than 2^31 and some programming languages may have difficulty/silent defects in interpreting it. But it has at most 52 significant bits, so a signed 64-bit integer or double-precision float type are safe for storing this value.
- `file_path` (`str`) – Optional. File path. Use https://api.telegram.org/file/bot<token>/<file_path> to get the file.

Результат
Instance of the class

Тип результата
telebot.types.File

class telebot.types.ForceReply(*selective: Optional[bool] = None*, *input_field_placeholder: Optional[str] = None*)

Базовые классы: *JsonSerializable*

Upon receiving a message with this object, Telegram clients will display a reply interface to the user (act as if the user has selected the bot's message and tapped „Reply"). This can be extremely useful if you want to create user-friendly step-by-step interfaces without having to sacrifice privacy mode.

Telegram Documentation: https://core.telegram.org/bots/api#forcereply

Параметры

- `force_reply` (`bool`) – Shows reply interface to the user, as if they manually selected the bot's message and tapped „Reply"
- `input_field_placeholder` (`str`) – Optional. The placeholder to be shown in the input field when the reply is active; 1-64 characters
- `selective` (`bool`) – Optional. Use this parameter if you want to force reply from specific users only. Targets: 1) users that are @mentioned in the text of the Message object; 2) if the bot's message is a reply (has reply_to_message_id), sender of the original message.

Результат
Instance of the class

Тип результата
telebot.types.ForceReply

class telebot.types.ForumTopic(*message_thread_id: int*, *name: str*, *icon_color: int*, *icon_custom_emoji_id: Optional[str] = None*)

Базовые классы: *JsonDeserializable*

This object represents a forum topic.

Telegram documentation: https://core.telegram.org/bots/api#forumtopic

Параметры

- `message_thread_id` (`int`) – Unique identifier of the forum topic
- `name` (`str`) – Name of the topic
- `icon_color` (`int`) – Color of the topic icon in RGB format
- `icon_custom_emoji_id` (`str`) – Optional. Unique identifier of the custom emoji shown as the topic icon

Результат
Instance of the class

Тип результата
telebot.types.ForumTopic

`class telebot.types.ForumTopicClosed`

Базовые классы: *JsonDeserializable*

This object represents a service message about a forum topic closed in the chat. Currently holds no information.

Telegram documentation: https://core.telegram.org/bots/api#forumtopicclosed

`class telebot.types.ForumTopicCreated(`*name: str*, *icon_color: int*, *icon_custom_emoji_id: Optional[str] = None*`)`

Базовые классы: *JsonDeserializable*

This object represents a service message about a new forum topic created in the chat.

Telegram documentation: https://core.telegram.org/bots/api#forumtopiccreated

Параметры

- `name` (`str`) – Name of the topic
- `icon_color` (`int`) – Color of the topic icon in RGB format
- `icon_custom_emoji_id` (`str`) – Optional. Unique identifier of the custom emoji shown as the topic icon

Результат
Instance of the class

Тип результата
telebot.types.ForumTopicCreated

`class telebot.types.ForumTopicReopened`

Базовые классы: *JsonDeserializable*

This object represents a service message about a forum topic reopened in the chat. Currently holds no information.

Telegram documentation: https://core.telegram.org/bots/api#forumtopicreopened

class telebot.types.Game(*title*, *description*, *photo*, *text=None*, *text_entities=None*, *animation=None*, ***kwargs*)

Базовые классы: *JsonDeserializable*

This object represents a game. Use BotFather to create and edit games, their short names will act as unique identifiers.

Telegram Documentation: https://core.telegram.org/bots/api#game

Параметры

- `title` (`str`) – Title of the game
- `description` (`str`) – Description of the game
- `photo` (`list` of *telebot.types.PhotoSize*) – Photo that will be displayed in the game message in chats.
- `text` (`str`) – Optional. Brief description of the game or high scores included in the game message. Can be automatically edited to include current high scores for the game when the bot calls setGameScore, or manually edited using editMessageText. 0-4096 characters.
- `text_entities` (`list` of *telebot.types.MessageEntity*) – Optional. Special entities that appear in text, such as usernames, URLs, bot commands, etc.
- `animation` (*telebot.types.Animation*) – Optional. Animation that will be displayed in the game message in chats. Upload via BotFather

Результат
Instance of the class

Тип результата
telebot.types.Game

classmethod parse_entities(*message_entity_array*)

Parse the message entity array into a list of MessageEntity objects

classmethod parse_photo(*photo_size_array*)

Parse the photo array into a list of PhotoSize objects

class telebot.types.GameHighScore(*position*, *user*, *score*, ***kwargs*)

Базовые классы: *JsonDeserializable*

This object represents one row of the high scores table for a game.

Telegram Documentation: https://core.telegram.org/bots/api#gamehighscore

Параметры

- `position` (`int`) – Position in high score table for the game
- `user` (*telebot.types.User*) – User
- `score` (`int`) – Score

Результат
Instance of the class

Тип результата
telebot.types.GameHighScore

class telebot.types.InlineKeyboardButton(*text*, *url=None*, *callback_data=None*, *web_app=None*, *switch_inline_query=None*, *switch_inline_query_current_chat=None*, *callback_game=None*, *pay=None*, *login_url=None*, ***kwargs*)

Базовые классы: *Dictionaryable*, *JsonSerializable*, *JsonDeserializable*

This object represents one button of an inline keyboard. You must use exactly one of the optional fields.

Telegram Documentation: https://core.telegram.org/bots/api#inlinekeyboardbutton

Параметры

- `text` (`str`) – Label text on the button
- `url` (`str`) – Optional. HTTP or tg:// URL to be opened when the button is pressed. Links tg://user?id=<user_id> can be used to mention a user by their ID without using a username, if this is allowed by their privacy settings.
- `callback_data` (`str`) – Optional. Data to be sent in a callback query to the bot when button is pressed, 1-64 bytes
- `web_app` (*telebot.types.WebAppInfo*) – Optional. Description of the Web App that will be launched when the user presses the button. The Web App will be able to send an arbitrary message on behalf of the user using the method answerWebAppQuery. Available only in private chats between a user and the bot.
- `login_url` (*telebot.types.LoginUrl*) – Optional. An HTTPS URL used to automatically authorize the user. Can be used as a replacement for the Telegram Login Widget.
- `switch_inline_query` (`str`) – Optional. 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. May be empty, in which case just the bot's username will be inserted.Note: This offers an easy way for users to start using your bot in inline mode when they are currently in a private chat with it. Especially useful when combined with switch_pm... actions - in this case the user will be automatically returned to the chat they switched from, skipping the chat selection screen.
- `switch_inline_query_current_chat` (`str`) – Optional. If set, pressing the button will insert the bot's username and the specified inline query in the current chat's input field. May be empty, in which case only the bot's username will be inserted.This offers a quick way for the user to open your bot in inline mode in the same chat - good for selecting something from multiple options.
- `callback_game` (`telebot.types.CallbackGame`) – Optional. Description of the game that will be launched when the user presses the button. NOTE: This type of button must always be the first button in the first row.
- `pay` (`bool`) – Optional. Specify True, to send a Pay button. NOTE: This type of button must always be the first button in the first row and can only be used in invoice messages.

Результат

Instance of the class

Тип результата

telebot.types.InlineKeyboardButton

class telebot.types.InlineKeyboardMarkup(*keyboard=None*, *row_width=3*)

Базовые классы: *Dictionaryable*, *JsonSerializable*, *JsonDeserializable*

This object represents an inline keyboard that appears right next to the message it belongs to.

Примечание: It is recommended to use *telebot.util.quick_markup()* instead.

Список 1: Example of a custom keyboard with buttons.

```
from telebot.util import quick_markup

markup = quick_markup(
    {'text': 'Press me', 'callback_data': 'press'},
    {'text': 'Press me too', 'callback_data': 'press_too'}
)
```

Telegram Documentation: https://core.telegram.org/bots/api#inlinekeyboardmarkup

Параметры

`inline_keyboard` (`list` of `list` of *telebot.types.InlineKeyboardButton*) – `list` of button rows, each represented by an `list` of *telebot.types.InlineKeyboardButton* objects

Результат

Instance of the class

Тип результата

telebot.types.InlineKeyboardMarkup

add(**args*, *row_width=None*)

This method adds buttons to the keyboard without exceeding row_width.

E.g. InlineKeyboardMarkup.add(«A», «B», «C») yields the json result: {keyboard: [[«A»], [«B»], [«C»]]} when row_width is set to 1. When row_width is set to 2, the result: {keyboard: [[«A», «B»], [«C»]]} See https://core.telegram.org/bots/api#inlinekeyboardmarkup

Параметры

- `args` (`list` of *telebot.types.InlineKeyboardButton*) – Array of InlineKeyboardButton to append to the keyboard
- `row_width` (`int`) – width of row

Результат

self, to allow function chaining.

Тип результата

telebot.types.InlineKeyboardMarkup

max_row_keys = 8

row(**args*)

Adds a list of InlineKeyboardButton to the keyboard. This method does not consider row_width.

InlineKeyboardMarkup.row(«A»).row(«B», «C»).to_json() outputs: „{keyboard: [[«A»], [«B», «C»]]}“ See https://core.telegram.org/bots/api#inlinekeyboardmarkup

Параметры

`args` (`list` of *telebot.types.InlineKeyboardButton*) – Array of InlineKeyboardButton to append to the keyboard

Результат
self, to allow function chaining.

Тип результата
telebot.types.InlineKeyboardMarkup

class telebot.types.InlineQuery(*id*, *from_user*, *query*, *offset*, *chat_type=None*, *location=None*, ***kwargs*)

Базовые классы: *JsonDeserializable*

This object represents an incoming inline query. When the user sends an empty query, your bot could return some default or trending results.

Telegram Documentation: https://core.telegram.org/bots/api#inlinequery

Параметры

- `id` (`str`) – Unique identifier for this query
- `from_user` (*telebot.types.User*) – Sender
- `query` (`str`) – Text of the query (up to 256 characters)
- `offset` (`str`) – Offset of the results to be returned, can be controlled by the bot
- `chat_type` (`str`) – Optional. Type of the chat from which the inline query was sent. Can be either “sender” for a private chat with the inline query sender, “private”, “group”, “supergroup”, or “channel”. The chat type should be always known for requests sent from official clients and most third-party clients, unless the request was sent from a secret chat
- `location` (*telebot.types.Location*) – Optional. Sender location, only for bots that request user location

Результат
Instance of the class

Тип результата
telebot.types.InlineQuery

class telebot.types.InlineQueryResultArticle(*id*, *title*, *input_message_content*, *reply_markup=None*, *url=None*, *hide_url=None*, *description=None*, *thumb_url=None*, *thumb_width=None*, *thumb_height=None*)

Базовые классы: *InlineQueryResultBase*

Represents a link to an article or web page.

Telegram Documentation: https://core.telegram.org/bots/api#inlinequeryresultarticle

Параметры

- `type` (`str`) – Type of the result, must be article
- `id` (`str`) – Unique identifier for this result, 1-64 Bytes
- `title` (`str`) – Title of the result
- `input_message_content` (`telebot.types.InputMessageContent`) – Content of the message to be sent
- `reply_markup` (*telebot.types.InlineKeyboardMarkup*) – Optional. Inline keyboard attached to the message
- `url` (`str`) – Optional. URL of the result

- `hide_url` (`bool`) – Optional. Pass True, if you don't want the URL to be shown in the message
- `description` (`str`) – Optional. Short description of the result
- `thumb_url` (`str`) – Optional. Url of the thumbnail for the result
- `thumb_width` (`int`) – Optional. Thumbnail width
- `thumb_height` (`int`) – Optional. Thumbnail height

Результат
Instance of the class

Тип результата
telebot.types.InlineQueryResultArticle

class telebot.types.InlineQueryResultAudio(*id*, *audio_url*, *title*, *caption=None*, *caption_entities=None*, *parse_mode=None*, *performer=None*, *audio_duration=None*, *reply_markup=None*, *input_message_content=None*)

Базовые классы: *InlineQueryResultBase*

Represents a link to an MP3 audio file. By default, this audio file will be sent by the user. Alternatively, you can use input_message_content to send a message with the specified content instead of the audio.

Telegram Documentation: https://core.telegram.org/bots/api#inlinequeryresultaudio

Параметры

- `type` (`str`) – Type of the result, must be audio
- `id` (`str`) – Unique identifier for this result, 1-64 bytes
- `audio_url` (`str`) – A valid URL for the audio file
- `title` (`str`) – Title
- `caption` (`str`) – Optional. Caption, 0-1024 characters after entities parsing
- `parse_mode` (`str`) – Optional. Mode for parsing entities in the audio caption. See formatting options for more details.
- `caption_entities` (`list of` *telebot.types.MessageEntity*) – Optional. List of special entities that appear in the caption, which can be specified instead of parse_mode
- `performer` (`str`) – Optional. Performer
- `audio_duration` (`int`) – Optional. Audio duration in seconds
- `reply_markup` (*telebot.types.InlineKeyboardMarkup*) – Optional. Inline keyboard attached to the message
- `input_message_content` (`telebot.types.InputMessageContent`) – Optional. Content of the message to be sent instead of the audio

Результат
Instance of the class

Тип результата
telebot.types.InlineQueryResultAudio

class telebot.types.InlineQueryResultBase(*type*, *id*, *title=None*, *caption=None*, *input_message_content=None*, *reply_markup=None*, *caption_entities=None*, *parse_mode=None*)

Базовые классы: ABC, *Dictionaryable*, *JsonSerializable*

This object represents one result of an inline query. Telegram clients currently support results of the following 20 types:

- *InlineQueryResultCachedAudio*
- *InlineQueryResultCachedDocument*
- *InlineQueryResultCachedGif*
- *InlineQueryResultCachedMpeg4Gif*
- *InlineQueryResultCachedPhoto*
- *InlineQueryResultCachedSticker*
- *InlineQueryResultCachedVideo*
- *InlineQueryResultCachedVoice*
- *InlineQueryResultArticle*
- *InlineQueryResultAudio*
- *InlineQueryResultContact*
- *InlineQueryResultGame*
- *InlineQueryResultDocument*
- *InlineQueryResultGif*
- *InlineQueryResultLocation*
- *InlineQueryResultMpeg4Gif*
- *InlineQueryResultPhoto*
- *InlineQueryResultVenue*
- *InlineQueryResultVideo*
- *InlineQueryResultVoice*

Telegram Documentation: https://core.telegram.org/bots/api#inlinequeryresult

class telebot.types.InlineQueryResultCachedAudio(*id*, *audio_file_id*, *caption=None*, *caption_entities=None*, *parse_mode=None*, *reply_markup=None*, *input_message_content=None*)

Базовые классы: *InlineQueryResultCachedBase*

Represents a link to an MP3 audio file stored on the Telegram servers. By default, this audio file will be sent by the user. Alternatively, you can use input_message_content to send a message with the specified content instead of the audio.

Telegram Documentation: https://core.telegram.org/bots/api#inlinequeryresultcachedaudio

Параметры

- type (str) – Type of the result, must be audio
- id (str) – Unique identifier for this result, 1-64 bytes

- `audio_file_id` (`str`) – A valid file identifier for the audio file
- `caption` (`str`) – Optional. Caption, 0-1024 characters after entities parsing
- `parse_mode` (`str`) – Optional. Mode for parsing entities in the audio caption. See formatting options for more details.
- `caption_entities` (`list` of *telebot.types.MessageEntity*) – Optional. List of special entities that appear in the caption, which can be specified instead of parse_mode
- `reply_markup` (*telebot.types.InlineKeyboardMarkup*) – Optional. Inline keyboard attached to the message
- `input_message_content` (`telebot.types.InputMessageContent`) – Optional. Content of the message to be sent instead of the audio

Результат
Instance of the class

Тип результата
telebot.types.InlineQueryResultCachedAudio

`class telebot.types.InlineQueryResultCachedBase`

Базовые классы: `ABC`, *JsonSerializable*

Base class of all InlineQueryResultCached* classes.

`class telebot.types.InlineQueryResultCachedDocument(`*id*, *document_file_id*, *title*, *description=None*, *caption=None*, *caption_entities=None*, *parse_mode=None*, *reply_markup=None*, *input_message_content=None*`)`

Базовые классы: *InlineQueryResultCachedBase*

Represents a link to a file stored on the Telegram servers. By default, this file will be sent by the user with an optional caption. Alternatively, you can use input_message_content to send a message with the specified content instead of the file.

Telegram Documentation: https://core.telegram.org/bots/api#inlinequeryresultcacheddocument

Параметры

- `type` (`str`) – Type of the result, must be document
- `id` (`str`) – Unique identifier for this result, 1-64 bytes
- `title` (`str`) – Title for the result
- `document_file_id` (`str`) – A valid file identifier for the file
- `description` (`str`) – Optional. Short description of the result
- `caption` (`str`) – Optional. Caption of the document to be sent, 0-1024 characters after entities parsing
- `parse_mode` (`str`) – Optional. Mode for parsing entities in the document caption. See formatting options for more details.
- `caption_entities` (`list` of *telebot.types.MessageEntity*) – Optional. List of special entities that appear in the caption, which can be specified instead of parse_mode

- `reply_markup` (*telebot.types.InlineKeyboardMarkup*) – Optional. Inline keyboard attached to the message
- `input_message_content` (`telebot.types.InputMessageContent`) – Optional. Content of the message to be sent instead of the file

Результат
Instance of the class

Тип результата
telebot.types.InlineQueryResultCachedDocument

`class telebot.types.InlineQueryResultCachedGif(`*id*, *gif_file_id*, *title=None*, *description=None*, *caption=None*, *caption_entities=None*, *parse_mode=None*, *reply_markup=None*, *input_message_content=None*`)`

Базовые классы: *InlineQueryResultCachedBase*

Represents a link to an animated GIF file stored on the Telegram servers. By default, this animated GIF file will be sent by the user with an optional caption. Alternatively, you can use input_message_content to send a message with specified content instead of the animation.

Telegram Documentation: https://core.telegram.org/bots/api#inlinequeryresultcachedgif

Параметры

- `type` (`str`) – Type of the result, must be gif
- `id` (`str`) – Unique identifier for this result, 1-64 bytes
- `gif_file_id` (`str`) – A valid file identifier for the GIF file
- `title` (`str`) – Optional. Title for the result
- `caption` (`str`) – Optional. Caption of the GIF file to be sent, 0-1024 characters after entities parsing
- `parse_mode` (`str`) – Optional. Mode for parsing entities in the caption. See formatting options for more details.
- `caption_entities` (`list` of *telebot.types.MessageEntity*) – Optional. List of special entities that appear in the caption, which can be specified instead of parse_mode
- `reply_markup` (*telebot.types.InlineKeyboardMarkup*) – Optional. Inline keyboard attached to the message
- `input_message_content` (`telebot.types.InputMessageContent`) – Optional. Content of the message to be sent instead of the GIF animation

Результат
Instance of the class

Тип результата
telebot.types.InlineQueryResultCachedGif

`class telebot.types.InlineQueryResultCachedMpeg4Gif(`*id*, *mpeg4_file_id*, *title=None*, *description=None*, *caption=None*, *caption_entities=None*, *parse_mode=None*, *reply_markup=None*, *input_message_content=None*`)`

Базовые классы: *InlineQueryResultCachedBase*

Represents a link to a video animation (H.264/MPEG-4 AVC video without sound) stored on the Telegram servers. By default, this animated MPEG-4 file will be sent by the user with an optional caption. Alternatively, you can use input_message_content to send a message with the specified content instead of the animation.

Telegram Documentation: https://core.telegram.org/bots/api#inlinequeryresultcachedmpeg4gif

Параметры

- `type` (`str`) – Type of the result, must be mpeg4_gif
- `id` (`str`) – Unique identifier for this result, 1-64 bytes
- `mpeg4_file_id` (`str`) – A valid file identifier for the MPEG4 file
- `title` (`str`) – Optional. Title for the result
- `caption` (`str`) – Optional. Caption of the MPEG-4 file to be sent, 0-1024 characters after entities parsing
- `parse_mode` (`str`) – Optional. Mode for parsing entities in the caption. See formatting options for more details.
- `caption_entities` (`list of` *telebot.types.MessageEntity*) – Optional. List of special entities that appear in the caption, which can be specified instead of parse_mode
- `reply_markup` (*telebot.types.InlineKeyboardMarkup*) – Optional. Inline keyboard attached to the message
- `input_message_content` (`telebot.types.InputMessageContent`) – Optional. Content of the message to be sent instead of the video animation

Результат

Instance of the class

Тип результата

telebot.types.InlineQueryResultCachedMpeg4Gif

`class telebot.types.InlineQueryResultCachedPhoto(`*id*, *photo_file_id*, *title=None*, *description=None*, *caption=None*, *caption_entities=None*, *parse_mode=None*, *reply_markup=None*, *input_message_content=None*`)`

Базовые классы: *InlineQueryResultCachedBase*

Represents a link to a photo stored on the Telegram servers. By default, this photo will be sent by the user with an optional caption. Alternatively, you can use input_message_content to send a message with the specified content instead of the photo.

Telegram Documentation: https://core.telegram.org/bots/api#inlinequeryresultcachedphoto

Параметры

- `type` (`str`) – Type of the result, must be photo
- `id` (`str`) – Unique identifier for this result, 1-64 bytes
- `photo_file_id` (`str`) – A valid file identifier of the photo
- `title` (`str`) – Optional. Title for the result
- `description` (`str`) – Optional. Short description of the result

- `caption` (`str`) – Optional. Caption of the photo to be sent, 0-1024 characters after entities parsing
- `parse_mode` (`str`) – Optional. Mode for parsing entities in the photo caption. See formatting options for more details.
- `caption_entities` (`list` of *telebot.types.MessageEntity*) – Optional. List of special entities that appear in the caption, which can be specified instead of parse_mode
- `reply_markup` (*telebot.types.InlineKeyboardMarkup*) – Optional. Inline keyboard attached to the message
- `input_message_content` (`telebot.types.InputMessageContent`) – Optional. Content of the message to be sent instead of the photo

Результат
Instance of the class

Тип результата
telebot.types.InlineQueryResultCachedPhoto

`class telebot.types.InlineQueryResultCachedSticker(id, sticker_file_id, reply_markup=None, input_message_content=None)`

Базовые классы: *InlineQueryResultCachedBase*

Represents a link to a sticker stored on the Telegram servers. By default, this sticker will be sent by the user. Alternatively, you can use input_message_content to send a message with the specified content instead of the sticker.

Telegram Documentation: https://core.telegram.org/bots/api#inlinequeryresultcachedsticker

Параметры

- `type` (`str`) – Type of the result, must be sticker
- `id` (`str`) – Unique identifier for this result, 1-64 bytes
- `sticker_file_id` (`str`) – A valid file identifier of the sticker
- `reply_markup` (*telebot.types.InlineKeyboardMarkup*) – Optional. Inline keyboard attached to the message
- `input_message_content` (`telebot.types.InputMessageContent`) – Optional. Content of the message to be sent instead of the sticker

Результат
Instance of the class

Тип результата
telebot.types.InlineQueryResultCachedSticker

`class telebot.types.InlineQueryResultCachedVideo(id, video_file_id, title, description=None, caption=None, caption_entities=None, parse_mode=None, reply_markup=None, input_message_content=None)`

Базовые классы: *InlineQueryResultCachedBase*

Represents a link to a video file stored on the Telegram servers. By default, this video file will be sent by the user with an optional caption. Alternatively, you can use input_message_content to send a message with the specified content instead of the video.

Telegram Documentation: https://core.telegram.org/bots/api#inlinequeryresultcachedvideo

Параметры

- `type` (`str`) – Type of the result, must be video
- `id` (`str`) – Unique identifier for this result, 1-64 bytes
- `video_file_id` (`str`) – A valid file identifier for the video file
- `title` (`str`) – Title for the result
- `description` (`str`) – Optional. Short description of the result
- `caption` (`str`) – Optional. Caption of the video to be sent, 0-1024 characters after entities parsing
- `parse_mode` (`str`) – Optional. Mode for parsing entities in the video caption. See formatting options for more details.
- `caption_entities` (`list` of *telebot.types.MessageEntity*) – Optional. List of special entities that appear in the caption, which can be specified instead of parse_mode
- `reply_markup` (*telebot.types.InlineKeyboardMarkup*) – Optional. Inline keyboard attached to the message
- `input_message_content` (`telebot.types.InputMessageContent`) – Optional. Content of the message to be sent instead of the video

Результат

Instance of the class

Тип результата

telebot.types.InlineQueryResultCachedVideo

`class telebot.types.InlineQueryResultCachedVoice(`*id*, *voice_file_id*, *title*, *caption=None*, *caption_entities=None*, *parse_mode=None*, *reply_markup=None*, *input_message_content=None*`)`

Базовые классы: *InlineQueryResultCachedBase*

Represents a link to a voice message stored on the Telegram servers. By default, this voice message will be sent by the user. Alternatively, you can use input_message_content to send a message with the specified content instead of the voice message.

Telegram Documentation: https://core.telegram.org/bots/api#inlinequeryresultcachedvoice

Параметры

- `type` (`str`) – Type of the result, must be voice
- `id` (`str`) – Unique identifier for this result, 1-64 bytes
- `voice_file_id` (`str`) – A valid file identifier for the voice message
- `title` (`str`) – Voice message title
- `caption` (`str`) – Optional. Caption, 0-1024 characters after entities parsing
- `parse_mode` (`str`) – Optional. Mode for parsing entities in the voice message caption. See formatting options for more details.
- `caption_entities` (`list` of *telebot.types.MessageEntity*) – Optional. List of special entities that appear in the caption, which can be specified instead of parse_mode

- `reply_markup` (*telebot.types.InlineKeyboardMarkup*) – Optional. Inline keyboard attached to the message
- `input_message_content` (`telebot.types.InputMessageContent`) – Optional. Content of the message to be sent instead of the voice message

Результат
Instance of the class

Тип результата
telebot.types.InlineQueryResultCachedVoice

`class telebot.types.InlineQueryResultContact(`*id*, *phone_number*, *first_name*, *last_name=None*, *vcard=None*, *reply_markup=None*, *input_message_content=None*, *thumb_url=None*, *thumb_width=None*, *thumb_height=None*`)`

Базовые классы: *InlineQueryResultBase*

Represents a contact with a phone number. By default, this contact will be sent by the user. Alternatively, you can use input_message_content to send a message with the specified content instead of the contact.

Telegram Documentation: https://core.telegram.org/bots/api#inlinequeryresultcontact

Параметры

- `type` (`str`) – Type of the result, must be contact
- `id` (`str`) – Unique identifier for this result, 1-64 Bytes
- `phone_number` (`str`) – Contact's phone number
- `first_name` (`str`) – Contact's first name
- `last_name` (`str`) – Optional. Contact's last name
- `vcard` (`str`) – Optional. Additional data about the contact in the form of a vCard, 0-2048 bytes
- `reply_markup` (*telebot.types.InlineKeyboardMarkup*) – Optional. Inline keyboard attached to the message
- `input_message_content` (`telebot.types.InputMessageContent`) – Optional. Content of the message to be sent instead of the contact
- `thumb_url` (`str`) – Optional. Url of the thumbnail for the result
- `thumb_width` (`int`) – Optional. Thumbnail width
- `thumb_height` (`int`) – Optional. Thumbnail height

Результат
Instance of the class

Тип результата
telebot.types.InlineQueryResultContact

`class telebot.types.InlineQueryResultDocument(`*id*, *title*, *document_url*, *mime_type*, *caption=None*, *caption_entities=None*, *parse_mode=None*, *description=None*, *reply_markup=None*, *input_message_content=None*, *thumb_url=None*, *thumb_width=None*, *thumb_height=None*`)`

Базовые классы: *InlineQueryResultBase*

Represents a link to a file. By default, this file will be sent by the user with an optional caption. Alternatively, you can use input_message_content to send a message with the specified content instead of the file. Currently, only .PDF and .ZIP files can be sent using this method.

Telegram Documentation: https://core.telegram.org/bots/api#inlinequeryresultdocument

Параметры

- `type` (`str`) – Type of the result, must be document
- `id` (`str`) – Unique identifier for this result, 1-64 bytes
- `title` (`str`) – Title for the result
- `caption` (`str`) – Optional. Caption of the document to be sent, 0-1024 characters after entities parsing
- `parse_mode` (`str`) – Optional. Mode for parsing entities in the document caption. See formatting options for more details.
- `caption_entities` (`list of telebot.types.MessageEntity`) – Optional. List of special entities that appear in the caption, which can be specified instead of parse_mode
- `document_url` (`str`) – A valid URL for the file
- `mime_type` (`str`) – MIME type of the content of the file, either "application/pdf" or "application/zip"
- `description` (`str`) – Optional. Short description of the result
- `reply_markup` (`telebot.types.InlineKeyboardMarkup`) – Optional. Inline keyboard attached to the message
- `input_message_content` (`telebot.types.InputMessageContent`) – Optional. Content of the message to be sent instead of the file
- `thumb_url` (`str`) – Optional. URL of the thumbnail (JPEG only) for the file
- `thumb_width` (`int`) – Optional. Thumbnail width
- `thumb_height` (`int`) – Optional. Thumbnail height

Результат
Instance of the class

Тип результата
telebot.types.InlineQueryResultDocument

`class telebot.types.InlineQueryResultGame(id, game_short_name, reply_markup=None)`

Базовые классы: *InlineQueryResultBase*

Represents a Game.

Telegram Documentation: https://core.telegram.org/bots/api#inlinequeryresultgame

Параметры

- `type` (`str`) – Type of the result, must be game
- `id` (`str`) – Unique identifier for this result, 1-64 bytes
- `game_short_name` (`str`) – Short name of the game
- `reply_markup` (`telebot.types.InlineKeyboardMarkup`) – Optional. Inline keyboard attached to the message

Результат
Instance of the class

Тип результата
telebot.types.InlineQueryResultGame

class telebot.types.InlineQueryResultGif(*id*, *gif_url*, *thumb_url*, *gif_width=None*, *gif_height=None*, *title=None*, *caption=None*, *caption_entities=None*, *reply_markup=None*, *input_message_content=None*, *gif_duration=None*, *parse_mode=None*, *thumb_mime_type=None*)

Базовые классы: *InlineQueryResultBase*

Represents a link to an animated GIF file. By default, this animated GIF file will be sent by the user with optional caption. Alternatively, you can use input_message_content to send a message with the specified content instead of the animation.

Telegram Documentation: https://core.telegram.org/bots/api#inlinequeryresultgif

Параметры

- `type` (`str`) – Type of the result, must be gif
- `id` (`str`) – Unique identifier for this result, 1-64 bytes
- `gif_url` (`str`) – A valid URL for the GIF file. File size must not exceed 1MB
- `gif_width` (`int`) – Optional. Width of the GIF
- `gif_height` (`int`) – Optional. Height of the GIF
- `gif_duration` (`int`) – Optional. Duration of the GIF in seconds
- `thumb_url` (`str`) – URL of the static (JPEG or GIF) or animated (MPEG4) thumbnail for the result
- `thumb_mime_type` (`str`) – Optional. MIME type of the thumbnail, must be one of “image/jpeg”, “image/gif”, or “video/mp4”. Defaults to “image/jpeg”
- `title` (`str`) – Optional. Title for the result
- `caption` (`str`) – Optional. Caption of the GIF file to be sent, 0-1024 characters after entities parsing
- `parse_mode` (`str`) – Optional. Mode for parsing entities in the caption. See formatting options for more details.
- `caption_entities` (`list of` *telebot.types.MessageEntity*) – Optional. List of special entities that appear in the caption, which can be specified instead of parse_mode
- `reply_markup` (*telebot.types.InlineKeyboardMarkup*) – Optional. Inline keyboard attached to the message
- `input_message_content` (`telebot.types.InputMessageContent`) – Optional. Content of the message to be sent instead of the GIF animation

Результат
Instance of the class

Тип результата
telebot.types.InlineQueryResultGif

class telebot.types.InlineQueryResultLocation(*id*, *title*, *latitude*, *longitude*, *horizontal_accuracy*, *live_period=None*, *reply_markup=None*, *input_message_content=None*, *thumb_url=None*, *thumb_width=None*, *thumb_height=None*, *heading=None*, *proximity_alert_radius=None*)

Базовые классы: *InlineQueryResultBase*

Represents a location on a map. By default, the location will be sent by the user. Alternatively, you can use input_message_content to send a message with the specified content instead of the location.

Telegram Documentation: https://core.telegram.org/bots/api#inlinequeryresultlocation

Параметры

- `type` (`str`) – Type of the result, must be location
- `id` (`str`) – Unique identifier for this result, 1-64 Bytes
- `latitude` (`float` number) – Location latitude in degrees
- `longitude` (`float` number) – Location longitude in degrees
- `title` (`str`) – Location title
- `horizontal_accuracy` (`float` number) – Optional. The radius of uncertainty for the location, measured in meters; 0-1500
- `live_period` (`int`) – Optional. Period in seconds for which the location can be updated, should be between 60 and 86400.
- `heading` (`int`) – Optional. For live locations, a direction in which the user is moving, in degrees. Must be between 1 and 360 if specified.
- `proximity_alert_radius` (`int`) – Optional. For live locations, a maximum distance for proximity alerts about approaching another chat member, in meters. Must be between 1 and 100000 if specified.
- `reply_markup` (*telebot.types.InlineKeyboardMarkup*) – Optional. Inline keyboard attached to the message
- `input_message_content` (`telebot.types.InputMessageContent`) – Optional. Content of the message to be sent instead of the location
- `thumb_url` (`str`) – Optional. Url of the thumbnail for the result
- `thumb_width` (`int`) – Optional. Thumbnail width
- `thumb_height` (`int`) – Optional. Thumbnail height

Результат

Instance of the class

Тип результата

telebot.types.InlineQueryResultLocation

class telebot.types.InlineQueryResultMpeg4Gif(*id*, *mpeg4_url*, *thumb_url*, *mpeg4_width=None*, *mpeg4_height=None*, *title=None*, *caption=None*, *caption_entities=None*, *parse_mode=None*, *reply_markup=None*, *input_message_content=None*, *mpeg4_duration=None*, *thumb_mime_type=None*)

Базовые классы: *InlineQueryResultBase*

Represents a link to a video animation (H.264/MPEG-4 AVC video without sound). By default, this animated MPEG-4 file will be sent by the user with optional caption. Alternatively, you can use input_message_content to send a message with the specified content instead of the animation.

Telegram Documentation: https://core.telegram.org/bots/api#inlinequeryresultmpeg4gif

Параметры

- `type` (`str`) – Type of the result, must be mpeg4_gif
- `id` (`str`) – Unique identifier for this result, 1-64 bytes
- `mpeg4_url` (`str`) – A valid URL for the MPEG4 file. File size must not exceed 1MB
- `mpeg4_width` (`int`) – Optional. Video width
- `mpeg4_height` (`int`) – Optional. Video height
- `mpeg4_duration` (`int`) – Optional. Video duration in seconds
- `thumb_url` (`str`) – URL of the static (JPEG or GIF) or animated (MPEG4) thumbnail for the result
- `thumb_mime_type` (`str`) – Optional. MIME type of the thumbnail, must be one of "image/jpeg", "image/gif", or "video/mp4". Defaults to "image/jpeg"
- `title` (`str`) – Optional. Title for the result
- `caption` (`str`) – Optional. Caption of the MPEG-4 file to be sent, 0-1024 characters after entities parsing
- `parse_mode` (`str`) – Optional. Mode for parsing entities in the caption. See formatting options for more details.
- `caption_entities` (`list` of *telebot.types.MessageEntity*) – Optional. List of special entities that appear in the caption, which can be specified instead of parse_mode
- `reply_markup` (*telebot.types.InlineKeyboardMarkup*) – Optional. Inline keyboard attached to the message
- `input_message_content` (`telebot.types.InputMessageContent`) – Optional. Content of the message to be sent instead of the video animation

Результат

Instance of the class

Тип результата

telebot.types.InlineQueryResultMpeg4Gif

`class telebot.types.InlineQueryResultPhoto(`*id*, *photo_url*, *thumb_url*, *photo_width=None*, *photo_height=None*, *title=None*, *description=None*, *caption=None*, *caption_entities=None*, *parse_mode=None*, *reply_markup=None*, *input_message_content=None*`)`

Базовые классы: *InlineQueryResultBase*

Represents a link to a photo. By default, this photo will be sent by the user with optional caption. Alternatively, you can use input_message_content to send a message with the specified content instead of the photo.

Telegram Documentation: https://core.telegram.org/bots/api#inlinequeryresultphoto

Параметры

- `type` (`str`) – Type of the result, must be photo
- `id` (`str`) – Unique identifier for this result, 1-64 bytes
- `photo_url` (`str`) – A valid URL of the photo. Photo must be in JPEG format. Photo size must not exceed 5MB
- `thumb_url` (`str`) – URL of the thumbnail for the photo
- `photo_width` (`int`) – Optional. Width of the photo
- `photo_height` (`int`) – Optional. Height of the photo
- `title` (`str`) – Optional. Title for the result
- `description` (`str`) – Optional. Short description of the result
- `caption` (`str`) – Optional. Caption of the photo to be sent, 0-1024 characters after entities parsing
- `parse_mode` (`str`) – Optional. Mode for parsing entities in the photo caption. See formatting options for more details.
- `caption_entities` (`list` of *telebot.types.MessageEntity*) – Optional. List of special entities that appear in the caption, which can be specified instead of parse_mode
- `reply_markup` (*telebot.types.InlineKeyboardMarkup*) – Optional. Inline keyboard attached to the message
- `input_message_content` (`telebot.types.InputMessageContent`) – Optional. Content of the message to be sent instead of the photo

Результат
Instance of the class

Тип результата
telebot.types.InlineQueryResultPhoto

`class telebot.types.InlineQueryResultVenue(`*id*, *title*, *latitude*, *longitude*, *address*, *foursquare_id=None*, *foursquare_type=None*, *reply_markup=None*, *input_message_content=None*, *thumb_url=None*, *thumb_width=None*, *thumb_height=None*, *google_place_id=None*, *google_place_type=None*`)`

Базовые классы: *InlineQueryResultBase*

Represents a venue. By default, the venue will be sent by the user. Alternatively, you can use input_message_content to send a message with the specified content instead of the venue.

Telegram Documentation: https://core.telegram.org/bots/api#inlinequeryresultvenue

Параметры

- `type` (`str`) – Type of the result, must be venue
- `id` (`str`) – Unique identifier for this result, 1-64 Bytes
- `latitude` (`float`) – Latitude of the venue location in degrees
- `longitude` (`float`) – Longitude of the venue location in degrees
- `title` (`str`) – Title of the venue
- `address` (`str`) – Address of the venue

- `foursquare_id` (`str`) – Optional. Foursquare identifier of the venue if known
- `foursquare_type` (`str`) – Optional. Foursquare type of the venue, if known. (For example, “arts_entertainment/default”, “arts_entertainment/aquarium” or “food/icecream”.)
- `google_place_id` (`str`) – Optional. Google Places identifier of the venue
- `google_place_type` (`str`) – Optional. Google Places type of the venue. (See supported types.)
- `reply_markup` (*telebot.types.InlineKeyboardMarkup*) – Optional. Inline keyboard attached to the message
- `input_message_content` (`telebot.types.InputMessageContent`) – Optional. Content of the message to be sent instead of the venue
- `thumb_url` (`str`) – Optional. Url of the thumbnail for the result
- `thumb_width` (`int`) – Optional. Thumbnail width
- `thumb_height` (`int`) – Optional. Thumbnail height

Результат
Instance of the class

Тип результата
telebot.types.InlineQueryResultVenue

class telebot.types.InlineQueryResultVideo(*id*, *video_url*, *mime_type*, *thumb_url*, *title*, *caption=None*, *caption_entities=None*, *parse_mode=None*, *video_width=None*, *video_height=None*, *video_duration=None*, *description=None*, *reply_markup=None*, *input_message_content=None*)

Базовые классы: *InlineQueryResultBase*

Represents a link to a page containing an embedded video player or a video file. By default, this video file will be sent by the user with an optional caption. Alternatively, you can use input_message_content to send a message with the specified content instead of the video.

Telegram Documentation: https://core.telegram.org/bots/api#inlinequeryresultvideo

Параметры

- `type` (`str`) – Type of the result, must be video
- `id` (`str`) – Unique identifier for this result, 1-64 bytes
- `video_url` (`str`) – A valid URL for the embedded video player or video file
- `mime_type` (`str`) – MIME type of the content of the video URL, “text/html” or “video/mp4”
- `thumb_url` (`str`) – URL of the thumbnail (JPEG only) for the video
- `title` (`str`) – Title for the result
- `caption` (`str`) – Optional. Caption of the video to be sent, 0-1024 characters after entities parsing
- `parse_mode` (`str`) – Optional. Mode for parsing entities in the video caption. See formatting options for more details.

- `caption_entities` (`list` of *telebot.types.MessageEntity*) – Optional. List of special entities that appear in the caption, which can be specified instead of parse_mode
- `video_width` (`int`) – Optional. Video width
- `video_height` (`int`) – Optional. Video height
- `video_duration` (`int`) – Optional. Video duration in seconds
- `description` (`str`) – Optional. Short description of the result
- `reply_markup` (*telebot.types.InlineKeyboardMarkup*) – Optional. Inline keyboard attached to the message
- `input_message_content` (`telebot.types.InputMessageContent`) – Optional. Content of the message to be sent instead of the video. This field is required if InlineQueryResultVideo is used to send an HTML-page as a result (e.g., a YouTube video).

Результат
Instance of the class

Тип результата
telebot.types.InlineQueryResultVideo

class telebot.types.InlineQueryResultVoice(*id*, *voice_url*, *title*, *caption=None*, *caption_entities=None*, *parse_mode=None*, *voice_duration=None*, *reply_markup=None*, *input_message_content=None*)

Базовые классы: *InlineQueryResultBase*

Represents a link to a voice recording in an .OGG container encoded with OPUS. By default, this voice recording will be sent by the user. Alternatively, you can use input_message_content to send a message with the specified content instead of the the voice message.

Telegram Documentation: https://core.telegram.org/bots/api#inlinequeryresultvoice

Параметры

- `type` (`str`) – Type of the result, must be voice
- `id` (`str`) – Unique identifier for this result, 1-64 bytes
- `voice_url` (`str`) – A valid URL for the voice recording
- `title` (`str`) – Recording title
- `caption` (`str`) – Optional. Caption, 0-1024 characters after entities parsing
- `parse_mode` (`str`) – Optional. Mode for parsing entities in the voice message caption. See formatting options for more details.
- `caption_entities` (`list` of *telebot.types.MessageEntity*) – Optional. List of special entities that appear in the caption, which can be specified instead of parse_mode
- `voice_duration` (`int`) – Optional. Recording duration in seconds
- `reply_markup` (*telebot.types.InlineKeyboardMarkup*) – Optional. Inline keyboard attached to the message
- `input_message_content` (`telebot.types.InputMessageContent`) – Optional. Content of the message to be sent instead of the voice recording

Результат
Instance of the class

Тип результата
telebot.types.InlineQueryResultVoice

class telebot.types.InputContactMessageContent(*phone_number*, *first_name*, *last_name=None*, *vcard=None*)

Базовые классы: *Dictionaryable*

Represents the content of a contact message to be sent as the result of an inline query.

Telegram Documentation: https://core.telegram.org/bots/api#inputcontactmessagecontent

Параметры

- `phone_number` (`str`) – Contact's phone number
- `first_name` (`str`) – Contact's first name
- `last_name` (`str`) – Optional. Contact's last name
- `vcard` (`str`) – Optional. Additional data about the contact in the form of a vCard, 0-2048 bytes

Результат
Instance of the class

Тип результата
telebot.types.InputContactMessageContent

class telebot.types.InputFile(*file*)

Базовые классы: `object`

A class to send files through Telegram Bot API.

You need to pass a file, which should be an instance of `io.IOBase` or `pathlib.Path`, or `str`.

If you pass an `str` as a file, it will be opened and closed by the class.

Параметры
`file` (`io.IOBase` or `pathlib.Path` or `str`) – A file to send.

Список 2: Example on sending a file using this class

```
from telebot.types import InputFile

# Sending a file from disk
bot.send_document(
    chat_id,
    InputFile('/path/to/file/file.txt')
)


# Sending a file from an io.IOBase object
with open('/path/to/file/file.txt', 'rb') as f:
    bot.send_document(
        chat_id,
        InputFile(f)
    )
```

(continues on next page)

(продолжение с предыдущей страницы)

```
# Sending a file using pathlib.Path:
bot.send_document(
    chat_id,
    InputFile(pathlib.Path('/path/to/file/file.txt'))
)
```

property file

File object.

class telebot.types.InputInvoiceMessageContent(*title*, *description*, *payload*, *provider_token*, *currency*, *prices*, *max_tip_amount=None*, *suggested_tip_amounts=None*, *provider_data=None*, *photo_url=None*, *photo_size=None*, *photo_width=None*, *photo_height=None*, *need_name=None*, *need_phone_number=None*, *need_email=None*, *need_shipping_address=None*, *send_phone_number_to_provider=None*, *send_email_to_provider=None*, *is_flexible=None*)

Базовые классы: *Dictionaryable*

Represents the content of an invoice message to be sent as the result of an inline query.

Telegram Documentation: https://core.telegram.org/bots/api#inputinvoicemessagecontent

Параметры

- `title` (`str`) – Product name, 1-32 characters
- `description` (`str`) – Product description, 1-255 characters
- `payload` (`str`) – Bot-defined invoice payload, 1-128 bytes. This will not be displayed to the user, use for your internal processes.
- `provider_token` (`str`) – Payment provider token, obtained via @BotFather
- `currency` (`str`) – Three-letter ISO 4217 currency code, see more on currencies
- `prices` (`list` of *telebot.types.LabeledPrice*) – Price breakdown, a JSON-serialized list of components (e.g. product price, tax, discount, delivery cost, delivery tax, bonus, etc.)
- `max_tip_amount` (`int`) – Optional. The maximum accepted amount for tips in the smallest units of the currency (integer, not float/double). For example, for a maximum tip of US$ 1.45 pass max_tip_amount = 145. See the exp parameter in currencies.json, it shows the number of digits past the decimal point for each currency (2 for the majority of currencies). Defaults to 0
- `suggested_tip_amounts` (`list` of `int`) – Optional. A JSON-serialized array of suggested amounts of tip in the smallest units of the currency (integer, not float/double). At most 4 suggested tip amounts can be specified. The suggested tip amounts must be positive, passed in a strictly increased order and must not exceed max_tip_amount.
- `provider_data` (`str`) – Optional. A JSON-serialized object for data about the invoice, which will be shared with the payment provider. A detailed description of the required fields should be provided by the payment provider.

- `photo_url` (`str`) – Optional. URL of the product photo for the invoice. Can be a photo of the goods or a marketing image for a service.
- `photo_size` (`int`) – Optional. Photo size in bytes
- `photo_width` (`int`) – Optional. Photo width
- `photo_height` (`int`) – Optional. Photo height
- `need_name` (`bool`) – Optional. Pass True, if you require the user's full name to complete the order
- `need_phone_number` (`bool`) – Optional. Pass True, if you require the user's phone number to complete the order
- `need_email` (`bool`) – Optional. Pass True, if you require the user's email address to complete the order
- `need_shipping_address` (`bool`) – Optional. Pass True, if you require the user's shipping address to complete the order
- `send_phone_number_to_provider` (`bool`) – Optional. Pass True, if the user's phone number should be sent to provider
- `send_email_to_provider` (`bool`) – Optional. Pass True, if the user's email address should be sent to provider
- `is_flexible` (`bool`) – Optional. Pass True, if the final price depends on the shipping method

Результат
Instance of the class

Тип результата
telebot.types.InputInvoiceMessageContent

`class telebot.types.InputLocationMessageContent(`*latitude*, *longitude*, *horizontal_accuracy=None*, *live_period=None*, *heading=None*, *proximity_alert_radius=None*`)`

Базовые классы: *Dictionaryable*

Represents the content of a location message to be sent as the result of an inline query.

Telegram Documentation: https://core.telegram.org/bots/api#inputlocationmessagecontent

Параметры

- `latitude` (`float`) – Latitude of the location in degrees
- `longitude` (`float`) – Longitude of the location in degrees
- `horizontal_accuracy` (`float` number) – Optional. The radius of uncertainty for the location, measured in meters; 0-1500
- `live_period` (`int`) – Optional. Period in seconds for which the location can be updated, should be between 60 and 86400.
- `heading` (`int`) – Optional. For live locations, a direction in which the user is moving, in degrees. Must be between 1 and 360 if specified.
- `proximity_alert_radius` (`int`) – Optional. For live locations, a maximum distance for proximity alerts about approaching another chat member, in meters. Must be between 1 and 100000 if specified.

Результат
Instance of the class

Тип результата
telebot.types.InputLocationMessageContent

class telebot.types.InputMedia(*type*, *media*, *caption=None*, *parse_mode=None*, *caption_entities=None*)

Базовые классы: *Dictionaryable*, *JsonSerializable*

This object represents the content of a media message to be sent. It should be one of

- *InputMediaAnimation*
- *InputMediaDocument*
- *InputMediaAudio*
- *InputMediaPhoto*
- *InputMediaVideo*

class telebot.types.InputMediaAnimation(*media*, *thumb=None*, *caption=None*, *parse_mode=None*, *width=None*, *height=None*, *duration=None*)

Базовые классы: *InputMedia*

Represents an animation file (GIF or H.264/MPEG-4 AVC video without sound) to be sent.

Telegram Documentation: https://core.telegram.org/bots/api#inputmediaanimation

Параметры

- `type` (`str`) – Type of the result, must be animation
- `media` (`str`) – File to send. Pass a file_id to send a file that exists on the Telegram servers (recommended), pass an HTTP URL for Telegram to get a file from the Internet, or pass "attach://<file_attach_name>" to upload a new one using multipart/form-data under <file_attach_name> name. More information on Sending Files »
- `thumb` (InputFile or `str`) – Optional. Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass "attach://<file_attach_name>" if the thumbnail was uploaded using multipart/form-data under <file_attach_name>. More information on Sending Files »
- `caption` (`str`) – Optional. Caption of the animation to be sent, 0-1024 characters after entities parsing
- `parse_mode` (`str`) – Optional. Mode for parsing entities in the animation caption. See formatting options for more details.
- `caption_entities` (`list` of *telebot.types.MessageEntity*) – Optional. List of special entities that appear in the caption, which can be specified instead of parse_mode
- `width` (`int`) – Optional. Animation width
- `height` (`int`) – Optional. Animation height
- `duration` (`int`) – Optional. Animation duration in seconds

Результат
Instance of the class

Тип результата
telebot.types.InputMediaAnimation

class telebot.types.InputMediaAudio(*media*, *thumb=None*, *caption=None*, *parse_mode=None*, *duration=None*, *performer=None*, *title=None*)

Базовые классы: *InputMedia*

Represents an audio file to be treated as music to be sent.

Telegram Documentation: https://core.telegram.org/bots/api#inputmediaaudio

Параметры

- `type` (`str`) – Type of the result, must be audio
- `media` (`str`) – File to send. Pass a file_id to send a file that exists on the Telegram servers (recommended), pass an HTTP URL for Telegram to get a file from the Internet, or pass "attach://<file_attach_name>" to upload a new one using multipart/form-data under <file_attach_name> name. More information on Sending Files »
- `thumb` (InputFile or `str`) – Optional. Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass "attach://<file_attach_name>" if the thumbnail was uploaded using multipart/form-data under <file_attach_name>. More information on Sending Files »
- `caption` (`str`) – Optional. Caption of the audio to be sent, 0-1024 characters after entities parsing
- `parse_mode` (`str`) – Optional. Mode for parsing entities in the audio caption. See formatting options for more details.
- `caption_entities` (`list` of *telebot.types.MessageEntity*) – Optional. List of special entities that appear in the caption, which can be specified instead of parse_mode
- `duration` (`int`) – Optional. Duration of the audio in seconds
- `performer` (`str`) – Optional. Performer of the audio
- `title` (`str`) – Optional. Title of the audio

Результат
Instance of the class

Тип результата
telebot.types.InputMediaAudio

class telebot.types.InputMediaDocument(*media*, *thumb=None*, *caption=None*, *parse_mode=None*, *disable_content_type_detection=None*)

Базовые классы: *InputMedia*

Represents a general file to be sent.

Telegram Documentation: https://core.telegram.org/bots/api#inputmediadocument

Параметры

- `type` (`str`) – Type of the result, must be document
- `media` (`str`) – File to send. Pass a file_id to send a file that exists on the Telegram servers (recommended), pass an HTTP URL for Telegram to get a file from the Internet, or pass "attach://<file_attach_name>" to upload a new one using multipart/form-data under <file_attach_name> name. More information on Sending Files »
- `thumb` (InputFile or `str`) – Optional. Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass "attach://<file_attach_name>" if the thumbnail was uploaded using multipart/form-data under <file_attach_name>. More information on Sending Files »
- `caption` (`str`) – Optional. Caption of the document to be sent, 0-1024 characters after entities parsing
- `parse_mode` (`str`) – Optional. Mode for parsing entities in the document caption. See formatting options for more details.
- `caption_entities` (`list` of *telebot.types.MessageEntity*) – Optional. List of special entities that appear in the caption, which can be specified instead of parse_mode
- `disable_content_type_detection` (`bool`) – Optional. Disables automatic server-side content type detection for files uploaded using multipart/form-data. Always True, if the document is sent as part of an album.

Результат

Instance of the class

Тип результата

telebot.types.InputMediaDocument

`class telebot.types.InputMediaPhoto(`*media*, *caption=None*, *parse_mode=None*`)`

Базовые классы: *InputMedia*

Represents a photo to be sent.

Telegram Documentation: https://core.telegram.org/bots/api#inputmediaphoto

Параметры

- `type` (`str`) – Type of the result, must be photo
- `media` (`str`) – File to send. Pass a file_id to send a file that exists on the Telegram servers (recommended), pass an HTTP URL for Telegram to get a file from the Internet, or pass "attach://<file_attach_name>" to upload a new one using multipart/form-data under <file_attach_name> name. More information on Sending Files »
- `caption` (`str`) – Optional. Caption of the photo to be sent, 0-1024 characters after entities parsing
- `parse_mode` (`str`) – Optional. Mode for parsing entities in the photo caption. See formatting options for more details.

- `caption_entities` (`list` of *telebot.types.MessageEntity*) – Optional. List of special entities that appear in the caption, which can be specified instead of parse_mode

Результат
Instance of the class

Тип результата
telebot.types.InputMediaPhoto

class telebot.types.InputMediaVideo(*media*, *thumb=None*, *caption=None*, *parse_mode=None*, *width=None*, *height=None*, *duration=None*, *supports_streaming=None*)

Базовые классы: *InputMedia*

Represents a video to be sent.

Telegram Documentation: https://core.telegram.org/bots/api#inputmediavideo

Параметры

- `type` (`str`) – Type of the result, must be video
- `media` (`str`) – File to send. Pass a file_id to send a file that exists on the Telegram servers (recommended), pass an HTTP URL for Telegram to get a file from the Internet, or pass "attach://<file_attach_name>" to upload a new one using multipart/form-data under <file_attach_name> name. More information on Sending Files »
- `thumb` (InputFile or `str`) – Optional. Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass "attach://<file_attach_name>" if the thumbnail was uploaded using multipart/form-data under <file_attach_name>. More information on Sending Files »
- `caption` (`str`) – Optional. Caption of the video to be sent, 0-1024 characters after entities parsing
- `parse_mode` (`str`) – Optional. Mode for parsing entities in the video caption. See formatting options for more details.
- `caption_entities` (`list` of *telebot.types.MessageEntity*) – Optional. List of special entities that appear in the caption, which can be specified instead of parse_mode
- `width` (`int`) – Optional. Video width
- `height` (`int`) – Optional. Video height
- `duration` (`int`) – Optional. Video duration in seconds
- `supports_streaming` (`bool`) – Optional. Pass True, if the uploaded video is suitable for streaming

Результат
Instance of the class

Тип результата
telebot.types.InputMediaVideo

class telebot.types.InputTextMessageContent(*message_text*, *parse_mode=None*, *entities=None*, *disable_web_page_preview=None*)

Базовые классы: *Dictionaryable*

Represents the content of a text message to be sent as the result of an inline query.

Telegram Documentation: https://core.telegram.org/bots/api#inputtextmessagecontent

Параметры

- `message_text` (`str`) – Text of the message to be sent, 1-4096 characters
- `parse_mode` (`str`) – Optional. Mode for parsing entities in the message text. See formatting options for more details.
- `entities` (`list` of *telebot.types.MessageEntity*) – Optional. List of special entities that appear in message text, which can be specified instead of parse_mode
- `disable_web_page_preview` (`bool`) – Optional. Disables link previews for links in the sent message

Результат
Instance of the class

Тип результата
telebot.types.InputTextMessageContent

class telebot.types.InputVenueMessageContent(*latitude*, *longitude*, *title*, *address*, *foursquare_id=None*, *foursquare_type=None*, *google_place_id=None*, *google_place_type=None*)

Базовые классы: *Dictionaryable*

Represents the content of a venue message to be sent as the result of an inline query.

Telegram Documentation: https://core.telegram.org/bots/api#inputvenuemessagecontent

Параметры

- `latitude` (`float`) – Latitude of the venue in degrees
- `longitude` (`float`) – Longitude of the venue in degrees
- `title` (`str`) – Name of the venue
- `address` (`str`) – Address of the venue
- `foursquare_id` (`str`) – Optional. Foursquare identifier of the venue, if known
- `foursquare_type` (`str`) – Optional. Foursquare type of the venue, if known. (For example, “arts_entertainment/default”, “arts_entertainment/aquarium” or “food/icecream”.)
- `google_place_id` (`str`) – Optional. Google Places identifier of the venue
- `google_place_type` (`str`) – Optional. Google Places type of the venue. (See supported types.)

Результат
Instance of the class

Тип результата
telebot.types.InputVenueMessageContent

class telebot.types.Invoice(*title*, *description*, *start_parameter*, *currency*, *total_amount*, ***kwargs*)

Базовые классы: *JsonDeserializable*

This object contains basic information about an invoice.

Telegram Documentation: https://core.telegram.org/bots/api#invoice

Параметры

- `title` (`str`) – Product name
- `description` (`str`) – Product description
- `start_parameter` (`str`) – Unique bot deep-linking parameter that can be used to generate this invoice
- `currency` (`str`) – Three-letter ISO 4217 currency code
- `total_amount` (`int`) – Total price in the smallest units of the currency (integer, not float/double). For example, for a price of US$ 1.45 pass amount = 145. See the exp parameter in currencies.json, it shows the number of digits past the decimal point for each currency (2 for the majority of currencies).

Результат
Instance of the class

Тип результата
telebot.types.Invoice

class telebot.types.JsonDeserializable

Базовые классы: `object`

Subclasses of this class are guaranteed to be able to be created from a json-style dict or json formatted string. All subclasses of this class must override de_json.

class telebot.types.JsonSerializable

Базовые классы: `object`

Subclasses of this class are guaranteed to be able to be converted to JSON format. All subclasses of this class must override to_json.

class telebot.types.KeyboardButton(*text: str*, *request_contact: Optional[bool] = None*, *request_location: Optional[bool] = None*, *request_poll: Optional[KeyboardButtonPollType] = None*, *web_app: Optional[WebAppInfo] = None*)

Базовые классы: *Dictionaryable*, *JsonSerializable*

This object represents one button of the reply keyboard. For simple text buttons String can be used instead of this object to specify text of the button. Optional fields web_app, request_contact, request_location, and request_poll are mutually exclusive.

Telegram Documentation: https://core.telegram.org/bots/api#keyboardbutton

Параметры

- `text` (`str`) – Text of the button. If none of the optional fields are used, it will be sent as a message when the button is pressed
- `request_contact` (`bool`) – Optional. If True, the user's phone number will be sent as a contact when the button is pressed. Available in private chats only.
- `request_location` (`bool`) – Optional. If True, the user's current location will be sent when the button is pressed. Available in private chats only.

- `request_poll` (*telebot.types.KeyboardButtonPollType*) – Optional. If specified, the user will be asked to create a poll and send it to the bot when the button is pressed. Available in private chats only.
- `web_app` (*telebot.types.WebAppInfo*) – Optional. If specified, the described Web App will be launched when the button is pressed. The Web App will be able to send a "web_app_data" service message. Available in private chats only.

Результат
Instance of the class

Тип результата
telebot.types.KeyboardButton

`class telebot.types.KeyboardButtonPollType(type='')`

Базовые классы: *Dictionaryable*

This object represents type of a poll, which is allowed to be created and sent when the corresponding button is pressed.

Telegram Documentation: https://core.telegram.org/bots/api#keyboardbuttonpolltype

Параметры
`type` (`str`) – Optional. If quiz is passed, the user will be allowed to create only polls in the quiz mode. If regular is passed, only regular polls will be allowed. Otherwise, the user will be allowed to create a poll of any type.

Результат
Instance of the class

Тип результата
telebot.types.KeyboardButtonPollType

`class telebot.types.LabeledPrice(label, amount)`

Базовые классы: *JsonSerializable*, *Dictionaryable*

This object represents a portion of the price for goods or services.

Telegram Documentation: https://core.telegram.org/bots/api#labeledprice

Параметры

- `label` (`str`) – Portion label
- `amount` (`int`) – Price of the product in the smallest units of the currency (integer, not float/double). For example, for a price of US$ 1.45 pass amount = 145. See the exp parameter in currencies.json, it shows the number of digits past the decimal point for each currency (2 for the majority of currencies).

Результат
Instance of the class

Тип результата
telebot.types.LabeledPrice

`class telebot.types.Location(longitude, latitude, horizontal_accuracy=None, live_period=None, heading=None, proximity_alert_radius=None, **kwargs)`

Базовые классы: *JsonDeserializable*, *JsonSerializable*, *Dictionaryable*

This object represents a point on the map.

Telegram Documentation: https://core.telegram.org/bots/api#location

Параметры

- `longitude` (`float`) – Longitude as defined by sender
- `latitude` (`float`) – Latitude as defined by sender
- `horizontal_accuracy` (`float` number) – Optional. The radius of uncertainty for the location, measured in meters; 0-1500
- `live_period` (`int`) – Optional. Time relative to the message sending date, during which the location can be updated; in seconds. For active live locations only.
- `heading` (`int`) – Optional. The direction in which user is moving, in degrees; 1-360. For active live locations only.
- `proximity_alert_radius` (`int`) – Optional. The maximum distance for proximity alerts about approaching another chat member, in meters. For sent live locations only.

Результат
Instance of the class

Тип результата
telebot.types.Location

class telebot.types.LoginUrl(*url*, *forward_text=None*, *bot_username=None*, *request_write_access=None*, ***kwargs*)

Базовые классы: *Dictionaryable*, *JsonSerializable*, *JsonDeserializable*

This object represents a parameter of the inline keyboard button used to automatically authorize a user. Serves as a great replacement for the Telegram Login Widget when the user is coming from Telegram. All the user needs to do is tap/click a button and confirm that they want to log in:

Telegram Documentation: https://core.telegram.org/bots/api#loginurl

Параметры

- `url` (`str`) – An HTTPS URL to be opened with user authorization data added to the query string when the button is pressed. If the user refuses to provide authorization data, the original URL without information about the user will be opened. The data added is the same as described in Receiving authorization data. NOTE: You must always check the hash of the received data to verify the authentication and the integrity of the data as described in Checking authorization.
- `forward_text` (`str`) – Optional. New text of the button in forwarded messages.
- `bot_username` (`str`) – Optional. Username of a bot, which will be used for user authorization. See Setting up a bot for more details. If not specified, the current bot's username will be assumed. The url's domain must be the same as the domain linked with the bot. See Linking your domain to the bot for more details.
- `request_write_access` (`bool`) – Optional. Pass True to request the permission for your bot to send messages to the user.

Результат
Instance of the class

Тип результата
telebot.types.LoginUrl

class telebot.types.MaskPosition(*point*, *x_shift*, *y_shift*, *scale*, ***kwargs*)

Базовые классы: *Dictionaryable*, *JsonDeserializable*, *JsonSerializable*

This object describes the position on faces where a mask should be placed by default.

Telegram Documentation: https://core.telegram.org/bots/api#maskposition

Параметры

- `point` (`str`) – The part of the face relative to which the mask should be placed. One of “forehead”, “eyes”, “mouth”, or “chin”.
- `x_shift` (`float` number) – Shift by X-axis measured in widths of the mask scaled to the face size, from left to right. For example, choosing -1.0 will place mask just to the left of the default mask position.
- `y_shift` (`float` number) – Shift by Y-axis measured in heights of the mask scaled to the face size, from top to bottom. For example, 1.0 will place the mask just below the default mask position.
- `scale` (`float` number) – Mask scaling coefficient. For example, 2.0 means double size.

Результат
Instance of the class

Тип результата
telebot.types.MaskPosition

`class telebot.types.MenuButton`

Базовые классы: *JsonDeserializable*, *JsonSerializable*, *Dictionaryable*

This object describes the bot’s menu button in a private chat. It should be one of

- *MenuButtonCommands*
- *MenuButtonWebApp*
- *MenuButtonDefault*

If a menu button other than MenuButtonDefault is set for a private chat, then it is applied in the chat. Otherwise the default menu button is applied. By default, the menu button opens the list of bot commands.

`class telebot.types.MenuButtonCommands(type)`

Базовые классы: *MenuButton*

Represents a menu button, which opens the bot’s list of commands.

Telegram Documentation: https://core.telegram.org/bots/api#menubuttoncommands

Параметры
`type` (`str`) – Type of the button, must be commands

Результат
Instance of the class

Тип результата
telebot.types.MenuButtonCommands

`class telebot.types.MenuButtonDefault(type)`

Базовые классы: *MenuButton*

Describes that no specific value for the menu button was set.

Telegram Documentation: https://core.telegram.org/bots/api#menubuttondefault

Параметры
`type` (`str`) – Type of the button, must be default

Результат
Instance of the class

Тип результата
telebot.types.MenuButtonDefault

class telebot.types.MenuButtonWebApp(*type*, *text*, *web_app*)

Базовые классы: *MenuButton*

Represents a menu button, which launches a Web App.

Telegram Documentation: https://core.telegram.org/bots/api#menubuttonwebapp

Параметры

- `type` (`str`) – Type of the button, must be web_app
- `text` (`str`) – Text on the button
- `web_app` (*telebot.types.WebAppInfo*) – Description of the Web App that will be launched when the user presses the button. The Web App will be able to send an arbitrary message on behalf of the user using the method answerWebAppQuery.

Результат
Instance of the class

Тип результата
telebot.types.MenuButtonWebApp

class telebot.types.Message(*message_id*, *from_user*, *date*, *chat*, *content_type*, *options*, *json_string*)

Базовые классы: *JsonDeserializable*

This object represents a message.

Telegram Documentation: https://core.telegram.org/bots/api#message

Параметры

- `message_id` (`int`) – Unique message identifier inside this chat
- `message_thread_id` (`int`) – Optional. Unique identifier of a message thread to which the message belongs; for supergroups only
- `from_user` (*telebot.types.User*) – Optional. Sender of the message; empty for messages sent to channels. For backward compatibility, the field contains a fake sender user in non-channel chats, if the message was sent on behalf of a chat.
- `sender_chat` (*telebot.types.Chat*) – Optional. Sender of the message, sent on behalf of a chat. For example, the channel itself for channel posts, the supergroup itself for messages from anonymous group administrators, the linked channel for messages automatically forwarded to the discussion group. For backward compatibility, the field from contains a fake sender user in non-channel chats, if the message was sent on behalf of a chat.
- `date` (`int`) – Date the message was sent in Unix time
- `chat` (*telebot.types.Chat*) – Conversation the message belongs to
- `forward_from` (*telebot.types.User*) – Optional. For forwarded messages, sender of the original message
- `forward_from_chat` (*telebot.types.Chat*) – Optional. For messages forwarded from channels or from anonymous administrators, information about the original sender chat

- `forward_from_message_id` (`int`) – Optional. For messages forwarded from channels, identifier of the original message in the channel
- `forward_signature` (`str`) – Optional. For forwarded messages that were originally sent in channels or by an anonymous chat administrator, signature of the message sender if present
- `forward_sender_name` (`str`) – Optional. Sender's name for messages forwarded from users who disallow adding a link to their account in forwarded messages
- `forward_date` (`int`) – Optional. For forwarded messages, date the original message was sent in Unix time
- `is_topic_message` (`bool`) – Optional. True, if the message is sent to a forum topic
- `is_automatic_forward` (`bool`) – Optional. `bool`, if the message is a channel post that was automatically forwarded to the connected discussion group
- `reply_to_message` (*telebot.types.Message*) – Optional. 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.
- `via_bot` (*telebot.types.User*) – Optional. Bot through which the message was sent
- `edit_date` (`int`) – Optional. Date the message was last edited in Unix time
- `has_protected_content` (`bool`) – Optional. `bool`, if the message can't be forwarded
- `media_group_id` (`str`) – Optional. The unique identifier of a media message group this message belongs to
- `author_signature` (`str`) – Optional. Signature of the post author for messages in channels, or the custom title of an anonymous group administrator
- `text` (`str`) – Optional. For text messages, the actual UTF-8 text of the message
- `entities` (`list` of *telebot.types.MessageEntity*) – Optional. For text messages, special entities like usernames, URLs, bot commands, etc. that appear in the text
- `animation` (*telebot.types.Animation*) – Optional. Message is an animation, information about the animation. For backward compatibility, when this field is set, the document field will also be set
- `audio` (*telebot.types.Audio*) – Optional. Message is an audio file, information about the file
- `document` (*telebot.types.Document*) – Optional. Message is a general file, information about the file
- `photo` (`list` of *telebot.types.PhotoSize*) – Optional. Message is a photo, available sizes of the photo
- `sticker` (*telebot.types.Sticker*) – Optional. Message is a sticker, information about the sticker
- `video` (*telebot.types.Video*) – Optional. Message is a video, information about the video
- `video_note` (*telebot.types.VideoNote*) – Optional. Message is a video note, information about the video message
- `voice` (*telebot.types.Voice*) – Optional. Message is a voice message, information about the file

- caption (str) – Optional. Caption for the animation, audio, document, photo, video or voice
- caption_entities (list of *telebot.types.MessageEntity*) – Optional. For messages with a caption, special entities like usernames, URLs, bot commands, etc. that appear in the caption
- contact (*telebot.types.Contact*) – Optional. Message is a shared contact, information about the contact
- dice (*telebot.types.Dice*) – Optional. Message is a dice with random value
- game (*telebot.types.Game*) – Optional. Message is a game, information about the game. More about games »
- poll (*telebot.types.Poll*) – Optional. Message is a native poll, information about the poll
- venue (*telebot.types.Venue*) – Optional. Message is a venue, information about the venue. For backward compatibility, when this field is set, the location field will also be set
- location (*telebot.types.Location*) – Optional. Message is a shared location, information about the location
- new_chat_members (list of *telebot.types.User*) – Optional. New members that were added to the group or supergroup and information about them (the bot itself may be one of these members)
- left_chat_member (*telebot.types.User*) – Optional. A member was removed from the group, information about them (this member may be the bot itself)
- new_chat_title (str) – Optional. A chat title was changed to this value
- new_chat_photo (list of *telebot.types.PhotoSize*) – Optional. A chat photo was change to this value
- delete_chat_photo (bool) – Optional. Service message: the chat photo was deleted
- group_chat_created (bool) – Optional. Service message: the group has been created
- supergroup_chat_created (bool) – Optional. Service message: the supergroup has been created. This field can't be received in a message coming through updates, because bot can't be a member of a supergroup when it is created. It can only be found in reply_to_message if someone replies to a very first message in a directly created supergroup.
- channel_chat_created (bool) – Optional. Service message: the channel has been created. This field can't be received in a message coming through updates, because bot can't be a member of a channel when it is created. It can only be found in reply_to_message if someone replies to a very first message in a channel.
- message_auto_delete_timer_changed (*telebot.types.MessageAutoDeleteTimerChanged*) – Optional. Service message: auto-delete timer settings changed in the chat
- migrate_to_chat_id (int) – Optional. The group has been migrated to a supergroup with the specified identifier. This number may have more than 32 significant bits and some programming languages may have difficulty/silent defects in interpreting it. But it has at most 52 significant bits, so a signed 64-bit integer or double-precision float type are safe for storing this identifier.

- migrate_from_chat_id (int) – Optional. The supergroup has been migrated from a group with the specified identifier. This number may have more than 32 significant bits and some programming languages may have difficulty/silent defects in interpreting it. But it has at most 52 significant bits, so a signed 64-bit integer or double-precision float type are safe for storing this identifier.
- pinned_message (*telebot.types.Message*) – Optional. Specified message was pinned. Note that the Message object in this field will not contain further reply_to_message fields even if it is itself a reply.
- invoice (*telebot.types.Invoice*) – Optional. Message is an invoice for a payment, information about the invoice. More about payments »
- successful_payment (*telebot.types.SuccessfulPayment*) – Optional. Message is a service message about a successful payment, information about the payment. More about payments »
- connected_website (str) – Optional. The domain name of the website on which the user has logged in. More about Telegram Login »
- passport_data (telebot.types.PassportData) – Optional. Telegram Passport data
- proximity_alert_triggered (*telebot.types.ProximityAlertTriggered*) – Optional. Service message. A user in the chat triggered another user's proximity alert while sharing Live Location.
- forum_topic_created (*telebot.types.ForumTopicCreated*) – Optional. Service message: forum topic created
- forum_topic_closed (*telebot.types.ForumTopicClosed*) – Optional. Service message: forum topic closed
- forum_topic_reopened (*telebot.types.ForumTopicReopened*) – Optional. Service message: forum topic reopened
- video_chat_scheduled (*telebot.types.VideoChatScheduled*) – Optional. Service message: video chat scheduled
- video_chat_started (*telebot.types.VideoChatStarted*) – Optional. Service message: video chat started
- video_chat_ended (*telebot.types.VideoChatEnded*) – Optional. Service message: video chat ended
- video_chat_participants_invited (*telebot.types.VideoChatParticipantsInvited*) – Optional. Service message: new participants invited to a video chat
- web_app_data (*telebot.types.WebAppData*) – Optional. Service message: data sent by a Web App
- reply_markup (*telebot.types.InlineKeyboardMarkup*) – Optional. Inline keyboard attached to the message. login_url buttons are represented as ordinary url buttons.

Результат
Instance of the class

Тип результата
telebot.types.Message

property html_caption

Returns html-rendered caption.

property html_text

Returns html-rendered text.

classmethod parse_chat(*chat*)

Parses chat.

classmethod parse_entities(*message_entity_array*)

Parses message entity array.

classmethod parse_photo(*photo_size_array*)

Parses photo array.

class telebot.types.MessageAutoDeleteTimerChanged(*message_auto_delete_time*, ***kwargs*)

Базовые классы: *JsonDeserializable*

This object represents a service message about a change in auto-delete timer settings.

Telegram Documentation: https://core.telegram.org/bots/api#messageautodeletetimerchanged

Параметры

message_auto_delete_time (int) – New auto-delete time for messages in the chat; in seconds

Результат

Instance of the class

Тип результата

telebot.types.MessageAutoDeleteTimerChanged

class telebot.types.MessageEntity(*type*, *offset*, *length*, *url=None*, *user=None*, *language=None*, *custom_emoji_id=None*, ***kwargs*)

Базовые классы: *Dictionaryable*, *JsonSerializable*, *JsonDeserializable*

This object represents one special entity in a text message. For example, hashtags, usernames, URLs, etc.

Telegram Documentation: https://core.telegram.org/bots/api#messageentity

Параметры

- type (str) – Type of the entity. Currently, can be “mention” (@username), “hashtag” (#hashtag), “cashtag” ($USD), “bot_command” (/start@jobs_bot), “url” (https://telegram.org), “email” (do-not-reply@telegram.org), “phone_number” (+1-212-555-0123), “bold” (bold text), “italic” (italic text), “underline” (underlined text), “strikethrough” (strikethrough text), “spoiler” (spoiler message), “code” (monowidth string), “pre” (monowidth block), “text_link” (for clickable text URLs), “text_mention” (for users without usernames), “custom_emoji” (for inline custom emoji stickers)
- offset (int) – Offset in UTF-16 code units to the start of the entity
- length (int) – Length of the entity in UTF-16 code units
- url (str) – Optional. For “text_link” only, URL that will be opened after user taps on the text
- user (*telebot.types.User*) – Optional. For “text_mention” only, the mentioned user

- `language` (`str`) – Optional. For “pre” only, the programming language of the entity text
- `custom_emoji_id` (`str`) – Optional. For “custom_emoji” only, unique identifier of the custom emoji. Use get_custom_emoji_stickers to get full information about the sticker.

Результат
Instance of the class

Тип результата
telebot.types.MessageEntity

`static to_list_of_dicts(`*entity_list*`)` → Optional[List[Dict]]

Converts a list of MessageEntity objects to a list of dictionaries.

`class telebot.types.MessageID(`*message_id*, ***kwargs*`)`

Базовые классы: *JsonDeserializable*

This object represents a unique message identifier.

Telegram Documentation: https://core.telegram.org/bots/api#messageid

Параметры
`message_id` (`int`) – Unique message identifier

Результат
Instance of the class

Тип результата
`telebot.types.MessageId`

`class telebot.types.OrderInfo(`*name=None*, *phone_number=None*, *email=None*, *shipping_address=None*, ***kwargs*`)`

Базовые классы: *JsonDeserializable*

This object represents information about an order.

Telegram Documentation: https://core.telegram.org/bots/api#orderinfo

Параметры

- `name` (`str`) – Optional. User name
- `phone_number` (`str`) – Optional. User’s phone number
- `email` (`str`) – Optional. User email
- `shipping_address` (*telebot.types.ShippingAddress*) – Optional. User shipping address

Результат
Instance of the class

Тип результата
telebot.types.OrderInfo

`class telebot.types.PhotoSize(`*file_id*, *file_unique_id*, *width*, *height*, *file_size=None*, ***kwargs*`)`

Базовые классы: *JsonDeserializable*

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

Telegram Documentation: https://core.telegram.org/bots/api#photosize

Параметры

- `file_id` (`str`) – Identifier for this file, which can be used to download or reuse the file
- `file_unique_id` (`str`) – 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.
- `width` (`int`) – Photo width
- `height` (`int`) – Photo height
- `file_size` (`int`) – Optional. File size in bytes

Результат
Instance of the class

Тип результата
telebot.types.PhotoSize

class telebot.types.Poll(*question*, *options*, *poll_id=None*, *total_voter_count=None*, *is_closed=None*, *is_anonymous=None*, *type=None*, *allows_multiple_answers=None*, *correct_option_id=None*, *explanation=None*, *explanation_entities=None*, *open_period=None*, *close_date=None*, *poll_type=None*, ***kwargs*)

Базовые классы: *JsonDeserializable*

This object contains information about a poll.

Telegram Documentation: https://core.telegram.org/bots/api#poll

Параметры

- `id` (`str`) – Unique poll identifier
- `question` (`str`) – Poll question, 1-300 characters
- `options` (`list` of *telebot.types.PollOption*) – List of poll options
- `total_voter_count` (`int`) – Total number of users that voted in the poll
- `is_closed` (`bool`) – True, if the poll is closed
- `is_anonymous` (`bool`) – True, if the poll is anonymous
- `type` (`str`) – Poll type, currently can be "regular" or "quiz"
- `allows_multiple_answers` (`bool`) – True, if the poll allows multiple answers
- `correct_option_id` (`int`) – Optional. 0-based identifier of the correct answer option. Available only for polls in the quiz mode, which are closed, or was sent (not forwarded) by the bot or to the private chat with the bot.
- `explanation` (`str`) – Optional. Text that is shown when a user chooses an incorrect answer or taps on the lamp icon in a quiz-style poll, 0-200 characters
- `explanation_entities` (`list` of *telebot.types.MessageEntity*) – Optional. Special entities like usernames, URLs, bot commands, etc. that appear in the explanation
- `open_period` (`int`) – Optional. Amount of time in seconds the poll will be active after creation
- `close_date` (`int`) – Optional. Point in time (Unix timestamp) when the poll will be automatically closed

Результат
Instance of the class

Тип результата
telebot.types.Poll

`add(option)`

Add an option to the poll.

Параметры
`option` (*telebot.types.PollOption* or `str`) – Option to add

`class telebot.types.PollAnswer(poll_id, user, option_ids, **kwargs)`

Базовые классы: *JsonSerializable*, *JsonDeserializable*, *Dictionaryable*

This object represents an answer of a user in a non-anonymous poll.

Telegram Documentation: https://core.telegram.org/bots/api#pollanswer

Параметры

- `poll_id` (`str`) – Unique poll identifier
- `user` (*telebot.types.User*) – The user, who changed the answer to the poll
- `option_ids` (`list` of `int`) – 0-based identifiers of answer options, chosen by the user. May be empty if the user retracted their vote.

Результат
Instance of the class

Тип результата
telebot.types.PollAnswer

`class telebot.types.PollOption(text, voter_count=0, **kwargs)`

Базовые классы: *JsonDeserializable*

This object contains information about one answer option in a poll.

Telegram Documentation: https://core.telegram.org/bots/api#polloption

Параметры

- `text` (`str`) – Option text, 1-100 characters
- `voter_count` (`int`) – Number of users that voted for this option

Результат
Instance of the class

Тип результата
telebot.types.PollOption

`class telebot.types.PreCheckoutQuery(id, from_user, currency, total_amount, invoice_payload, shipping_option_id=None, order_info=None, **kwargs)`

Базовые классы: *JsonDeserializable*

This object contains information about an incoming pre-checkout query.

Telegram Documentation: https://core.telegram.org/bots/api#precheckoutquery

Параметры

- `id` (`str`) – Unique query identifier
- `from` (*telebot.types.User*) – User who sent the query
- `currency` (`str`) – Three-letter ISO 4217 currency code

- `total_amount` (`int`) – Total price in the smallest units of the currency (integer, not float/double). For example, for a price of US$ 1.45 pass amount = 145. See the exp parameter in currencies.json, it shows the number of digits past the decimal point for each currency (2 for the majority of currencies).
- `invoice_payload` (`str`) – Bot specified invoice payload
- `shipping_option_id` (`str`) – Optional. Identifier of the shipping option chosen by the user
- `order_info` (*telebot.types.OrderInfo*) – Optional. Order information provided by the user

Результат
Instance of the class

Тип результата
telebot.types.PreCheckoutQuery

`class telebot.types.ProximityAlertTriggered(`*traveler*, *watcher*, *distance*, ***kwargs*`)`

Базовые классы: *JsonDeserializable*

This object represents the content of a service message, sent whenever a user in the chat triggers a proximity alert set by another user.

Telegram Documentation: https://core.telegram.org/bots/api#proximityalerttriggered

Параметры

- `traveler` (*telebot.types.User*) – User that triggered the alert
- `watcher` (*telebot.types.User*) – User that set the alert
- `distance` (`int`) – The distance between the users

Результат
Instance of the class

Тип результата
telebot.types.ProximityAlertTriggered

`class telebot.types.ReplyKeyboardMarkup(`*resize_keyboard: Optional[bool] = None*, *one_time_keyboard: Optional[bool] = None*, *selective: Optional[bool] = None*, *row_width: int = 3*, *input_field_placeholder: Optional[str] = None*`)`

Базовые классы: *JsonSerializable*

This object represents a custom keyboard with reply options (see Introduction to bots for details and examples).

Список 3: Example on creating ReplyKeyboardMarkup object

```
from telebot.types import ReplyKeyboardMarkup, KeyboardButton

markup = ReplyKeyboardMarkup(resize_keyboard=True)
markup.add(KeyboardButton('Text'))
# or:
markup.add('Text')

# display this markup:
bot.send_message(chat_id, 'Text', reply_markup=markup)
```

Telegram Documentation: https://core.telegram.org/bots/api#replykeyboardmarkup

Параметры

- `keyboard` (`list` of `list` of *telebot.types.KeyboardButton*) – `list` of button rows, each represented by an `list` of *telebot.types.KeyboardButton* objects
- `resize_keyboard` (`bool`) – Optional. Requests clients to resize the keyboard vertically for optimal fit (e.g., make the keyboard smaller if there are just two rows of buttons). Defaults to false, in which case the custom keyboard is always of the same height as the app's standard keyboard.
- `one_time_keyboard` (`bool`) – Optional. Requests clients to hide the keyboard as soon as it's been used. The keyboard will still be available, but clients will automatically display the usual letter-keyboard in the chat - the user can press a special button in the input field to see the custom keyboard again. Defaults to false.
- `input_field_placeholder` (`str`) – Optional. The placeholder to be shown in the input field when the keyboard is active; 1-64 characters
- `selective` (`bool`) – Optional. Use this parameter if you want to show the keyboard to specific users only. Targets: 1) users that are @mentioned in the text of the Message object; 2) if the bot's message is a reply (has reply_to_message_id), sender of the original message.Example: A user requests to change the bot's language, bot replies to the request with a keyboard to select the new language. Other users in the group don't see the keyboard.

Результат

Instance of the class

Тип результата

telebot.types.ReplyKeyboardMarkup

`add(*args, row_width=None)`

This function adds strings to the keyboard, while not exceeding row_width. E.g. ReplyKeyboardMarkup#add(«A», «B», «C») yields the json result {keyboard: [[«A»], [«B»], [«C»]]} when row_width is set to 1. When row_width is set to 2, the following is the result of this function: {keyboard: [[«A», «B»], [«C»]]} See https://core.telegram.org/bots/api#replykeyboardmarkup

Параметры

- `args` (`str` or *telebot.types.KeyboardButton*) – KeyboardButton to append to the keyboard
- `row_width` (`int`) – width of row

Результат

self, to allow function chaining.

Тип результата

telebot.types.ReplyKeyboardMarkup

`max_row_keys = 12`

`row(*args)`

Adds a list of KeyboardButton to the keyboard. This function does not consider row_width. ReplyKeyboardMarkup#row(«A»)#row(«B», «C»)#to_json() outputs „{keyboard: [[«A»], [«B», «C»]]}“ See https://core.telegram.org/bots/api#replykeyboardmarkup

Параметры

`args` (`str`) – strings

Результат
self, to allow function chaining.

Тип результата
telebot.types.ReplyKeyboardMarkup

class telebot.types.ReplyKeyboardRemove(*selective=None*)

Базовые классы: *JsonSerializable*

Upon receiving a message with this object, Telegram clients will remove the current custom keyboard and display the default letter-keyboard. By default, custom keyboards are displayed until a new keyboard is sent by a bot. An exception is made for one-time keyboards that are hidden immediately after the user presses a button (see ReplyKeyboardMarkup).

Telegram Documentation: https://core.telegram.org/bots/api#replykeyboardremove

Параметры

- `remove_keyboard` (`bool`) – Requests clients to remove the custom keyboard (user will not be able to summon this keyboard; if you want to hide the keyboard from sight but keep it accessible, use one_time_keyboard in ReplyKeyboardMarkup) Note that this parameter is set to True by default by the library. You cannot modify it.
- `selective` (`bool`) – Optional. Use this parameter if you want to remove the keyboard for specific users only. Targets: 1) users that are @mentioned in the text of the Message object; 2) if the bot's message is a reply (has reply_to_message_id), sender of the original message.Example: A user votes in a poll, bot returns confirmation message in reply to the vote and removes the keyboard for that user, while still showing the keyboard with poll options to users who haven't voted yet.

Результат
Instance of the class

Тип результата
telebot.types.ReplyKeyboardRemove

class telebot.types.SentWebAppMessage(*inline_message_id=None*)

Базовые классы: *JsonDeserializable*, *Dictionaryable*

Describes an inline message sent by a Web App on behalf of a user.

Telegram Documentation: https://core.telegram.org/bots/api#sentwebappmessage

Параметры
`inline_message_id` (`str`) – Optional. Identifier of the sent inline message. Available only if there is an inline keyboard attached to the message.

Результат
Instance of the class

Тип результата
telebot.types.SentWebAppMessage

class telebot.types.ShippingAddress(*country_code*, *state*, *city*, *street_line1*, *street_line2*, *post_code*, ***kwargs*)

Базовые классы: *JsonDeserializable*

This object represents a shipping address.

Telegram Documentation: https://core.telegram.org/bots/api#shippingaddress

Параметры

- `country_code` (`str`) – Two-letter ISO 3166-1 alpha-2 country code
- `state` (`str`) – State, if applicable
- `city` (`str`) – City
- `street_line1` (`str`) – First line for the address
- `street_line2` (`str`) – Second line for the address
- `post_code` (`str`) – Address post code

Результат
Instance of the class

Тип результата
telebot.types.ShippingAddress

`class telebot.types.ShippingOption(id, title)`

Базовые классы: *JsonSerializable*

This object represents one shipping option.

Telegram Documentation: https://core.telegram.org/bots/api#shippingoption

Параметры

- `id` (`str`) – Shipping option identifier
- `title` (`str`) – Option title
- `prices` (`list` of *telebot.types.LabeledPrice*) – List of price portions

Результат
Instance of the class

Тип результата
telebot.types.ShippingOption

`add_price(*args)`

Add LabeledPrice to ShippingOption

Параметры
`args` (*LabeledPrice*) – LabeledPrices

Результат
None

`class telebot.types.ShippingQuery(id, from_user, invoice_payload, shipping_address, **kwargs)`

Базовые классы: *JsonDeserializable*

This object contains information about an incoming shipping query.

Telegram Documentation: https://core.telegram.org/bots/api#shippingquery

Параметры

- `id` (`str`) – Unique query identifier
- `from` (*telebot.types.User*) – User who sent the query
- `invoice_payload` (`str`) – Bot specified invoice payload
- `shipping_address` (*telebot.types.ShippingAddress*) – User specified shipping address

Результат
Instance of the class

Тип результата
telebot.types.ShippingQuery

class telebot.types.Sticker(*file_id*, *file_unique_id*, *type*, *width*, *height*, *is_animated*, *is_video*, *thumb=None*, *emoji=None*, *set_name=None*, *mask_position=None*, *file_size=None*, *premium_animation=None*, *custom_emoji_id=None*, ***kwargs*)

Базовые классы: *JsonDeserializable*

This object represents a sticker.

Telegram Documentation: https://core.telegram.org/bots/api#sticker

Параметры

- `file_id` (`str`) – Identifier for this file, which can be used to download or reuse the file
- `file_unique_id` (`str`) – 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`) – Type of the sticker, currently one of "regular", "mask", "custom_emoji". The type of the sticker is independent from its format, which is determined by the fields is_animated and is_video.
- `width` (`int`) – Sticker width
- `height` (`int`) – Sticker height
- `is_animated` (`bool`) – True, if the sticker is animated
- `is_video` (`bool`) – True, if the sticker is a video sticker
- `thumb` (*telebot.types.PhotoSize*) – Optional. Sticker thumbnail in the .WEBP or .JPG format
- `emoji` (`str`) – Optional. Emoji associated with the sticker
- `set_name` (`str`) – Optional. Name of the sticker set to which the sticker belongs
- `premium_animation` (*telebot.types.File*) – Optional. Premium animation for the sticker, if the sticker is premium
- `mask_position` (*telebot.types.MaskPosition*) – Optional. For mask stickers, the position where the mask should be placed
- `custom_emoji_id` (`str`) – Optional. For custom emoji stickers, unique identifier of the custom emoji
- `file_size` (`int`) – Optional. File size in bytes

Результат
Instance of the class

Тип результата
telebot.types.Sticker

class telebot.types.StickerSet(*name*, *title*, *sticker_type*, *is_animated*, *is_video*, *stickers*, *thumb=None*, ***kwargs*)

Базовые классы: *JsonDeserializable*

This object represents a sticker set.

Telegram Documentation: https://core.telegram.org/bots/api#stickerset

Параметры

- `name` (`str`) – Sticker set name
- `title` (`str`) – Sticker set title
- `sticker_type` (`str`) – Type of stickers in the set, currently one of "regular", "mask", "custom_emoji"
- `is_animated` (`bool`) – True, if the sticker set contains animated stickers
- `is_video` (`bool`) – True, if the sticker set contains video stickers
- `contains_masks` (`bool`) – True, if the sticker set contains masks. Deprecated since Bot API 6.2, use sticker_type instead.
- `stickers` (`list` of *telebot.types.Sticker*) – List of all set stickers
- `thumb` (*telebot.types.PhotoSize*) – Optional. Sticker set thumbnail in the .WEBP, .TGS, or .WEBM format

Результат
Instance of the class

Тип результата
telebot.types.StickerSet

`property contains_masks`

Deprecated since Bot API 6.2, use sticker_type instead.

`class telebot.types.SuccessfulPayment(`*currency*, *total_amount*, *invoice_payload*, *shipping_option_id=None*, *order_info=None*, *telegram_payment_charge_id=None*, *provider_payment_charge_id=None*, ***kwargs*`)`

Базовые классы: *JsonDeserializable*

This object contains basic information about a successful payment.

Telegram Documentation: https://core.telegram.org/bots/api#successfulpayment

Параметры

- `currency` (`str`) – Three-letter ISO 4217 currency code
- `total_amount` (`int`) – Total price in the smallest units of the currency (integer, not float/double). For example, for a price of US$ 1.45 pass amount = 145. See the exp parameter in currencies.json, it shows the number of digits past the decimal point for each currency (2 for the majority of currencies).
- `invoice_payload` (`str`) – Bot specified invoice payload
- `shipping_option_id` (`str`) – Optional. Identifier of the shipping option chosen by the user
- `order_info` (*telebot.types.OrderInfo*) – Optional. Order information provided by the user
- `telegram_payment_charge_id` (`str`) – Telegram payment identifier
- `provider_payment_charge_id` (`str`) – Provider payment identifier

Результат
Instance of the class

Тип результата
telebot.types.SuccessfulPayment

class telebot.types.Update(*update_id*, *message*, *edited_message*, *channel_post*, *edited_channel_post*, *inline_query*, *chosen_inline_result*, *callback_query*, *shipping_query*, *pre_checkout_query*, *poll*, *poll_answer*, *my_chat_member*, *chat_member*, *chat_join_request*)

Базовые классы: *JsonDeserializable*

This object represents an incoming update.At most one of the optional parameters can be present in any given update.

Telegram Documentation: https://core.telegram.org/bots/api#update

Параметры

- `update_id` (`int`) – 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.
- `message` (*telebot.types.Message*) – Optional. New incoming message of any kind - text, photo, sticker, etc.
- `edited_message` (*telebot.types.Message*) – Optional. New version of a message that is known to the bot and was edited
- `channel_post` (*telebot.types.Message*) – Optional. New incoming channel post of any kind - text, photo, sticker, etc.
- `edited_channel_post` (*telebot.types.Message*) – Optional. New version of a channel post that is known to the bot and was edited
- `inline_query` (*telebot.types.InlineQuery*) – Optional. New incoming inline query
- `chosen_inline_result` (*telebot.types.ChosenInlineResult*) – Optional. The result of an inline query that was chosen by a user and sent to their chat partner. Please see our documentation on the feedback collecting for details on how to enable these updates for your bot.
- `callback_query` (*telebot.types.CallbackQuery*) – Optional. New incoming callback query
- `shipping_query` (*telebot.types.ShippingQuery*) – Optional. New incoming shipping query. Only for invoices with flexible price
- `pre_checkout_query` (*telebot.types.PreCheckoutQuery*) – Optional. New incoming pre-checkout query. Contains full information about checkout
- `poll` (*telebot.types.Poll*) – Optional. New poll state. Bots receive only updates about stopped polls and polls, which are sent by the bot
- `poll_answer` (*telebot.types.PollAnswer*) – Optional. A user changed their answer in a non-anonymous poll. Bots receive new votes only in polls that were sent by the bot itself.
- `my_chat_member` (*telebot.types.ChatMemberUpdated*) – Optional. The bot's chat member status was updated in a chat. For private chats, this update is received only when the bot is blocked or unblocked by the user.

- `chat_member` (*telebot.types.ChatMemberUpdated*) – Optional. A chat member's status was updated in a chat. The bot must be an administrator in the chat and must explicitly specify "chat_member" in the list of allowed_updates to receive these updates.
- `chat_join_request` (*telebot.types.ChatJoinRequest*) – Optional. A request to join the chat has been sent. The bot must have the can_invite_users administrator right in the chat to receive these updates.

Результат
Instance of the class

Тип результата
telebot.types.Update

class telebot.types.User(*id*, *is_bot*, *first_name*, *last_name=None*, *username=None*, *language_code=None*, *can_join_groups=None*, *can_read_all_group_messages=None*, *supports_inline_queries=None*, *is_premium=None*, *added_to_attachment_menu=None*, ***kwargs*)

Базовые классы: *JsonDeserializable*, *Dictionaryable*, *JsonSerializable*

This object represents a Telegram user or bot.

Telegram Documentation: https://core.telegram.org/bots/api#user

Параметры

- `id` (`int`) – Unique identifier for this user or bot. This number may have more than 32 significant bits and some programming languages may have difficulty/silent defects in interpreting it. But it has at most 52 significant bits, so a 64-bit integer or double-precision float type are safe for storing this identifier.
- `is_bot` (`bool`) – True, if this user is a bot
- `first_name` (`str`) – User's or bot's first name
- `last_name` (`str`) – Optional. User's or bot's last name
- `username` (`str`) – Optional. User's or bot's username
- `language_code` (`str`) – Optional. IETF language tag of the user's language
- `is_premium` (`bool`) – Optional. `bool`, if this user is a Telegram Premium user
- `added_to_attachment_menu` (`bool`) – Optional. `bool`, if this user added the bot to the attachment menu
- `can_join_groups` (`bool`) – Optional. True, if the bot can be invited to groups. Returned only in getMe.
- `can_read_all_group_messages` (`bool`) – Optional. True, if privacy mode is disabled for the bot. Returned only in getMe.
- `supports_inline_queries` (`bool`) – Optional. True, if the bot supports inline queries. Returned only in getMe.

Результат
Instance of the class

Тип результата
telebot.types.User

property full_name

User's full name

Type
return

class telebot.types.UserProfilePhotos(*total_count*, *photos=None*, ***kwargs*)

Базовые классы: *JsonDeserializable*

This object represent a user's profile pictures.

Telegram Documentation: https://core.telegram.org/bots/api#userprofilephotos

Параметры

- **total_count** (int) – Total number of profile pictures the target user has
- **photos** (list of list of *telebot.types.PhotoSize*) – Requested profile pictures (in up to 4 sizes each)

Результат
Instance of the class

Тип результата
telebot.types.UserProfilePhotos

class telebot.types.Venue(*location*, *title*, *address*, *foursquare_id=None*, *foursquare_type=None*, *google_place_id=None*, *google_place_type=None*, ***kwargs*)

Базовые классы: *JsonDeserializable*

This object represents a venue.

Telegram Documentation: https://core.telegram.org/bots/api#venue

Параметры

- **location** (*telebot.types.Location*) – Venue location. Can't be a live location
- **title** (str) – Name of the venue
- **address** (str) – Address of the venue
- **foursquare_id** (str) – Optional. Foursquare identifier of the venue
- **foursquare_type** (str) – Optional. Foursquare type of the venue. (For example, "arts_entertainment/default", "arts_entertainment/aquarium" or "food/icecream".)
- **google_place_id** (str) – Optional. Google Places identifier of the venue
- **google_place_type** (str) – Optional. Google Places type of the venue. (See supported types.)

Результат
Instance of the class

Тип результата
telebot.types.Venue

class telebot.types.Video(*file_id*, *file_unique_id*, *width*, *height*, *duration*, *thumb=None*, *file_name=None*, *mime_type=None*, *file_size=None*, ***kwargs*)

Базовые классы: *JsonDeserializable*

This object represents a video file.

Telegram Documentation: https://core.telegram.org/bots/api#video

Параметры

- `file_id` (`str`) – Identifier for this file, which can be used to download or reuse the file
- `file_unique_id` (`str`) – 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.
- `width` (`int`) – Video width as defined by sender
- `height` (`int`) – Video height as defined by sender
- `duration` (`int`) – Duration of the video in seconds as defined by sender
- `thumb` (*telebot.types.PhotoSize*) – Optional. Video thumbnail
- `file_name` (`str`) – Optional. Original filename as defined by sender
- `mime_type` (`str`) – Optional. MIME type of the file as defined by sender
- `file_size` (`int`) – Optional. File size in bytes. It can be bigger than 2^31 and some programming languages may have difficulty/silent defects in interpreting it. But it has at most 52 significant bits, so a signed 64-bit integer or double-precision float type are safe for storing this value.

Результат
Instance of the class

Тип результата
telebot.types.Video

`class telebot.types.VideoChatEnded(`*duration*, ***kwargs*`)`

Базовые классы: *JsonDeserializable*

This object represents a service message about a video chat ended in the chat.

Telegram Documentation: https://core.telegram.org/bots/api#videochatended

Параметры
`duration` (`int`) – Video chat duration in seconds

Результат
Instance of the class

Тип результата
telebot.types.VideoChatEnded

`class telebot.types.VideoChatParticipantsInvited(`*users=None*, ***kwargs*`)`

Базовые классы: *JsonDeserializable*

This object represents a service message about new members invited to a video chat.

Telegram Documentation: https://core.telegram.org/bots/api#videochatparticipantsinvited

Параметры
`users` (`list` of *telebot.types.User*) – New members that were invited to the video chat

Результат
Instance of the class

Тип результата
telebot.types.VideoChatParticipantsInvited

class telebot.types.VideoChatScheduled(*start_ date*, ***kwargs*)

Базовые классы: *JsonDeserializable*

This object represents a service message about a video chat scheduled in the chat.

Telegram Documentation: https://core.telegram.org/bots/api#videochatscheduled

Параметры
`start_date` (`int`) – Point in time (Unix timestamp) when the video chat is supposed to be started by a chat administrator

Результат
Instance of the class

Тип результата
telebot.types.VideoChatScheduled

class telebot.types.VideoChatStarted

Базовые классы: *JsonDeserializable*

This object represents a service message about a video chat started in the chat. Currently holds no information.

class telebot.types.VideoNote(*file_ id*, *file_ unique_ id*, *length*, *duration*, *thumb=None*, *file_ size=None*, ***kwargs*)

Базовые классы: *JsonDeserializable*

This object represents a video message (available in Telegram apps as of v.4.0).

Telegram Documentation: https://core.telegram.org/bots/api#videonote

Параметры

- `file_id` (`str`) – Identifier for this file, which can be used to download or reuse the file
- `file_unique_id` (`str`) – 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.
- `length` (`int`) – Video width and height (diameter of the video message) as defined by sender
- `duration` (`int`) – Duration of the video in seconds as defined by sender
- `thumb` (*telebot.types.PhotoSize*) – Optional. Video thumbnail
- `file_size` (`int`) – Optional. File size in bytes

Результат
Instance of the class

Тип результата
telebot.types.VideoNote

class telebot.types.Voice(*file_ id*, *file_ unique_ id*, *duration*, *mime_ type=None*, *file_ size=None*, ***kwargs*)

Базовые классы: *JsonDeserializable*

This object represents a voice note.

Telegram Documentation: https://core.telegram.org/bots/api#voice

Параметры

- `file_id` (`str`) – Identifier for this file, which can be used to download or reuse the file
- `file_unique_id` (`str`) – 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.
- `duration` (`int`) – Duration of the audio in seconds as defined by sender
- `mime_type` (`str`) – Optional. MIME type of the file as defined by sender
- `file_size` (`int`) – Optional. File size in bytes. It can be bigger than 2^31 and some programming languages may have difficulty/silent defects in interpreting it. But it has at most 52 significant bits, so a signed 64-bit integer or double-precision float type are safe for storing this value.

Результат
Instance of the class

Тип результата
telebot.types.Voice

`class telebot.types.VoiceChatEnded(*args, **kwargs)`

Базовые классы: *VideoChatEnded*

Deprecated, use *VideoChatEnded* instead.

`class telebot.types.VoiceChatParticipantsInvited(*args, **kwargs)`

Базовые классы: *VideoChatParticipantsInvited*

Deprecated, use *VideoChatParticipantsInvited* instead.

`class telebot.types.VoiceChatScheduled(*args, **kwargs)`

Базовые классы: *VideoChatScheduled*

Deprecated, use *VideoChatScheduled* instead.

`class telebot.types.VoiceChatStarted`

Базовые классы: *VideoChatStarted*

Deprecated, use *VideoChatStarted* instead.

`class telebot.types.WebAppData(data, button_text)`

Базовые классы: *JsonDeserializable*, *Dictionaryable*

Describes data sent from a Web App to the bot.

Telegram Documentation: https://core.telegram.org/bots/api#webappdata

Параметры

- `data` (`str`) – The data. Be aware that a bad client can send arbitrary data in this field.
- `button_text` (`str`) – Text of the web_app keyboard button from which the Web App was opened. Be aware that a bad client can send arbitrary data in this field.

Результат
Instance of the class

Тип результата
telebot.types.WebAppData

class telebot.types.WebAppInfo(*url*, ***kwargs*)

Базовые классы: *JsonDeserializable*, *Dictionaryable*

Describes a Web App.

Telegram Documentation: https://core.telegram.org/bots/api#webappinfo

Параметры

`url` (`str`) – An HTTPS URL of a Web App to be opened with additional data as specified in Initializing Web Apps

Результат

Instance of the class

Тип результата

telebot.types.WebAppInfo

class telebot.types.WebhookInfo(*url*, *has_custom_certificate*, *pending_update_count*, *ip_address=None*, *last_error_date=None*, *last_error_message=None*, *last_synchronization_error_date=None*, *max_connections=None*, *allowed_updates=None*, ***kwargs*)

Базовые классы: *JsonDeserializable*

Describes the current status of a webhook.

Telegram Documentation: https://core.telegram.org/bots/api#webhookinfo

Параметры

- `url` (`str`) – Webhook URL, may be empty if webhook is not set up
- `has_custom_certificate` (`bool`) – True, if a custom certificate was provided for webhook certificate checks
- `pending_update_count` (`int`) – Number of updates awaiting delivery
- `ip_address` (`str`) – Optional. Currently used webhook IP address
- `last_error_date` (`int`) – Optional. Unix time for the most recent error that happened when trying to deliver an update via webhook
- `last_error_message` (`str`) – Optional. Error message in human-readable format for the most recent error that happened when trying to deliver an update via webhook
- `last_synchronization_error_date` (`int`) – Optional. Unix time of the most recent error that happened when trying to synchronize available updates with Telegram datacenters
- `max_connections` (`int`) – Optional. The maximum allowed number of simultaneous HTTPS connections to the webhook for update delivery
- `allowed_updates` (`list of str`) – Optional. A list of update types the bot is subscribed to. Defaults to all update types except chat_member

Результат

Instance of the class

Тип результата

telebot.types.WebhookInfo

1.3.4 TeleBot version

TeleBot methods

class telebot.ExceptionHandler

Базовые классы: object

Class for handling exceptions while Polling

handle(*exception*)

class telebot.Handler(*callback*, **args*, ***kwargs*)

Базовые классы: object

Class for (next step|reply) handlers.

telebot.REPLY_MARKUP_TYPES

telebot

Type
Module

alias of Union[*InlineKeyboardMarkup*, *ReplyKeyboardMarkup*, *ReplyKeyboardRemove*, *ForceReply*]

class telebot.TeleBot(*token: str*, *parse_mode: ~typing.Optional[str] = None*, *threaded: ~typing.Optional[bool] = True*, *skip_pending: ~typing.Optional[bool] = False*, *num_threads: ~typing.Optional[int] = 2*, *next_step_backend: ~typing.Optional[~telebot.handler_backends.HandlerBackend] = None*, *reply_backend: ~typing.Optional[~telebot.handler_backends.HandlerBackend] = None*, *exception_handler: ~typing.Optional[~telebot.ExceptionHandler] = None*, *last_update_id: ~typing.Optional[int] = 0*, *suppress_middleware_excepions: ~typing.Optional[bool] = False*, *state_storage: ~typing.Optional[~telebot.storage.base_storage.StateStorageBase] = <telebot.storage.memory_storage.StateMemoryStorage object>*, *use_class_middlewares: ~typing.Optional[bool] = False*, *disable_web_page_preview: ~typing.Optional[bool] = None*, *disable_notification: ~typing.Optional[bool] = None*, *protect_content: ~typing.Optional[bool] = None*, *allow_sending_without_reply: ~typing.Optional[bool] = None*, *colorful_logs: ~typing.Optional[bool] = False*)

Базовые классы: object

This is the main synchronous class for Bot.

It allows you to add handlers for different kind of updates.

Usage:

Список 4: Creating instance of TeleBot

```
from telebot import TeleBot
bot = TeleBot('token') # get token from @BotFather
# now you can register other handlers/update listeners,
# and use bot methods.
```

See more examples in examples/ directory: https://github.com/eternnoir/pyTelegramBotAPI/tree/master/examples

Примечание: Install coloredlogs module to specify colorful_logs=True

Параметры

- `token` (`str`) – Token of a bot, should be obtained from @BotFather
- `parse_mode` (`str`, optional) – Default parse mode, defaults to None
- `threaded` (`bool`, optional) – Threaded or not, defaults to True
- `skip_pending` (`bool`, optional) – Skips pending updates, defaults to False
- `num_threads` (`int`, optional) – Number of maximum parallel threads, defaults to 2
- `next_step_backend` (`telebot.handler_backends.HandlerBackend`, optional) – Next step backend class, defaults to None
- `reply_backend` (`telebot.handler_backends.HandlerBackend`, optional) – Reply step handler class, defaults to None
- `exception_handler` (*telebot.ExceptionHandler*, optional) – Exception handler to handle errors, defaults to None
- `last_update_id` (`int`, optional) – Last update's id, defaults to 0
- `suppress_middleware_excepions` (`bool`, optional) – Supress middleware exceptions, defaults to False
- `state_storage` (`telebot.storage.StateStorageBase`, optional) – Storage for states, defaults to StateMemoryStorage()
- `use_class_middlewares` (`bool`, optional) – Use class middlewares, defaults to False
- `disable_web_page_preview` (`bool`, optional) – Default value for disable_web_page_preview, defaults to None
- `disable_notification` (`bool`, optional) – Default value for disable_notification, defaults to None
- `protect_content` (`bool`, optional) – Default value for protect_content, defaults to None
- `allow_sending_without_reply` (`bool`, optional) – Default value for allow_sending_without_reply, defaults to None
- `colorful_logs` (`bool`, optional) – Outputs colorful logs

`add_custom_filter`(*custom_filter: Union[*SimpleCustomFilter, AdvancedCustomFilter*]*)

Create custom filter.

Список 5: Example on checking the text of a message

```
class TextMatchFilter(AdvancedCustomFilter):
    key = 'text'

    def check(self, message, text):
        return text == message.text
```

Параметры

- `custom_filter` – Class with check(message) method.

- `custom_filter` – Custom filter class with key.

`add_data`(*user_id: int*, *chat_id: Optional[int] = None*, ***kwargs*)

Add data to states.

Параметры

- `user_id` (`int`) – User's identifier
- `chat_id` (`int`) – Chat's identifier
- `kwargs` – Data to add

Результат

None

`add_sticker_to_set`(*user_id: int*, *name: str*, *emojis: str*, *png_sticker: Optional[Union[str, Any]] = None*, *tgs_sticker: Optional[Union[str, Any]] = None*, *webm_sticker: Optional[Union[str, Any]] = None*, *mask_position: Optional[MaskPosition] = None*) → bool

Use this method to add a new sticker to a set created by the bot. It's required to pass *png_sticker* or *tgs_sticker*. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#addstickertoset

Параметры

- `user_id` (`int`) – User identifier of created sticker set owner
- `name` (`str`) – Sticker set name
- `emojis` (`str`) – One or more emoji corresponding to the sticker
- `png_sticker` (`str` or `filelike object`) – PNG image with the sticker, must be up to 512 kilobytes in size, dimensions must not exceed 512px, and either width or height must be exactly 512px. Pass a file_id as a String to send a file that already exists on the Telegram servers, pass an HTTP URL as a String for Telegram to get a file from the Internet, or upload a new one using multipart/form-data.
- `tgs_sticker` (`str` or `filelike object`) – TGS animation with the sticker, uploaded using multipart/form-data.
- `webm_sticker` (`str` or `filelike object`) – WebM animation with the sticker, uploaded using multipart/form-data.
- `mask_position` (*`telebot.types.MaskPosition`*) – A JSON-serialized object for position where the mask should be placed on faces

Результат

On success, True is returned.

Тип результата

`bool`

`answer_callback_query`(*callback_query_id: int*, *text: Optional[str] = None*, *show_alert: Optional[bool] = None*, *url: Optional[str] = None*, *cache_time: Optional[int] = None*) → bool

Use this method to send answers to callback queries sent from inline keyboards. The answer will be displayed to the user as a notification at the top of the chat screen or as an alert.

Telegram documentation: https://core.telegram.org/bots/api#answercallbackquery

Параметры

- `callback_query_id` (`int`) – Unique identifier for the query to be answered
- `text` (`str`) – Text of the notification. If not specified, nothing will be shown to the user, 0-200 characters
- `show_alert` (`bool`) – If True, an alert will be shown by the client instead of a notification at the top of the chat screen. Defaults to false.
- `url` (`str`) – URL that will be opened by the user's client. If you have created a Game and accepted the conditions via @BotFather, specify the URL that opens your game - note that this will only work if the query comes from a callback_game button.
- `cache_time` – The maximum amount of time in seconds that the result of the callback query may be cached client-side. Telegram apps will support caching starting in version 3.14. Defaults to 0.

Результат
On success, True is returned.

Тип результата
`bool`

`answer_inline_query`(*inline_query_id: str*, *results: List[Any]*, *cache_time: Optional[int] = None*, *is_personal: Optional[bool] = None*, *next_offset: Optional[str] = None*, *switch_pm_text: Optional[str] = None*, *switch_pm_parameter: Optional[str] = None*) → bool

Use this method to send answers to an inline query. On success, True is returned. No more than 50 results per query are allowed.

Telegram documentation: https://core.telegram.org/bots/api#answerinlinequery

Параметры

- `inline_query_id` (`str`) – Unique identifier for the answered query
- `results` (`list` of `types.InlineQueryResult`) – Array of results for the inline query
- `cache_time` (`int`) – The maximum amount of time in seconds that the result of the inline query may be cached on the server.
- `is_personal` (`bool`) – Pass True, if results may be cached on the server side only for the user that sent the query.
- `next_offset` (`str`) – Pass the offset that a client should send in the next query with the same text to receive more results.
- `switch_pm_parameter` (`str`) – Deep-linking parameter for the /start message sent to the bot when user presses the switch button. 1-64 characters, only A-Z, a-z, 0-9, _ and - are allowed. Example: An inline bot that sends YouTube videos can ask the user to connect the bot to their YouTube account to adapt search results accordingly. To do this, it displays a „Connect your YouTube account" button above the results, or even before showing any. The user presses the button, switches to a private chat with the bot and, in doing so, passes a start parameter that instructs the bot to return an OAuth link. Once done, the bot can offer a switch_inline button so that the user can easily return to the chat where they wanted to use the bot's inline capabilities.
- `switch_pm_text` (`str`) – Parameter for the start message sent to the bot when user presses the switch button

Результат
On success, True is returned.

Тип результата
`bool`

`answer_pre_checkout_query`(*pre_checkout_query_id: int*, *ok: bool*, *error_message: Optional[str] = None*) → bool

Once the user has confirmed their payment and shipping details, the Bot API sends the final confirmation in the form of an Update with the field pre_checkout_query. Use this method to respond to such pre-checkout queries. On success, True is returned.

Примечание: The Bot API must receive an answer within 10 seconds after the pre-checkout query was sent.

Telegram documentation: https://core.telegram.org/bots/api#answerprecheckoutquery

Параметры

- `pre_checkout_query_id` (`int`) – Unique identifier for the query to be answered
- `ok` (`bool`) – Specify True if everything is alright (goods are available, etc.) and the bot is ready to proceed with the order. Use False if there are any problems.
- `error_message` (`str`) – Required if ok is False. Error message in human readable form that explains the reason for failure to proceed with the checkout (e.g. «Sorry, somebody just bought the last of our amazing black T-shirts while you were busy filling out your payment details. Please choose a different color or garment!»). Telegram will display this message to the user.

Результат
On success, True is returned.

Тип результата
`bool`

`answer_shipping_query`(*shipping_query_id: str*, *ok: bool*, *shipping_options: Optional[List[ShippingOption]] = None*, *error_message: Optional[str] = None*) → bool

Asks for an answer to a shipping question.

Telegram documentation: https://core.telegram.org/bots/api#answershippingquery

Параметры

- `shipping_query_id` (`str`) – Unique identifier for the query to be answered
- `ok` (`bool`) – Specify True if delivery to the specified address is possible and False if there are any problems (for example, if delivery to the specified address is not possible)
- `shipping_options` (`list` of `ShippingOption`) – Required if ok is True. A JSON-serialized array of available shipping options.
- `error_message` (`str`) – Required if ok is False. Error message in human readable form that explains why it is impossible to complete the order (e.g. «Sorry, delivery to your desired address is unavailable“). Telegram will display this message to the user.

Результат
On success, True is returned.

Тип результата
`bool`

`answer_web_app_query`(*web_app_query_id: str*, *result:* InlineQueryResultBase) → *SentWebAppMessage*

Use this method to set the result of an interaction with a Web App and send a corresponding message on behalf of the user to the chat from which the query originated. On success, a SentWebAppMessage object is returned.

Telegram Documentation: https://core.telegram.org/bots/api#answerwebappquery

Параметры

- `web_app_query_id` (`str`) – Unique identifier for the query to be answered
- `result` (*`telebot.types.InlineQueryResultBase`*) – A JSON-serialized object describing the message to be sent

Результат
On success, a SentWebAppMessage object is returned.

Тип результата
`telebot.types.SentWebAppMessage`

`approve_chat_join_request`(*chat_id: Union[str, int]*, *user_id: Union[int, str]*) → bool

Use this method to approve a chat join request. The bot must be an administrator in the chat for this to work and must have the can_invite_users administrator right. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#approvechatjoinrequest

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target supergroup (in the format @supergroupusername)
- `user_id` (`int` or `str`) – Unique identifier of the target user

Результат
True on success.

Тип результата
`bool`

`ban_chat_member`(*chat_id: Union[int, str]*, *user_id: int*, *until_date: Optional[Union[int, datetime]] = None*, *revoke_messages: Optional[bool] = None*) → bool

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

Telegram documentation: https://core.telegram.org/bots/api#banchatmember

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target group or username of the target supergroup or channel (in the format @channelusername)
- `user_id` (`int`) – Unique identifier of the target user
- `until_date` (`int` or `datetime`) – Date when the user will be unbanned, unix time. If user is banned for more than 366 days or less than 30 seconds from the current time they are considered to be banned forever

- `revoke_messages` (`bool`) – Bool: Pass True to delete all messages from the chat for the user that is being removed. If False, the user will be able to see messages in the group that were sent before the user was removed. Always True for supergroups and channels.

Результат
Returns True on success.

Тип результата
`bool`

`ban_chat_sender_chat(`*chat_id: Union[int, str]*, *sender_chat_id: Union[int, str]*`)` → bool

Use this method to ban a channel chat in a supergroup or a channel. The owner of the chat will not be able to send messages and join live streams on behalf of the chat, unless it is unbanned first. The bot must be an administrator in the supergroup or channel for this to work and must have the appropriate administrator rights. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#banchatsenderchat

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `sender_chat_id` (`int` or `str`) – Unique identifier of the target sender chat

Результат
True on success.

Тип результата
`bool`

`callback_query_handler(`*func*, ***kwargs*`)`

Handles new incoming callback query. As a parameter to the decorator function, it passes *`telebot.types.CallbackQuery`* object.

Параметры

- `func` (`function`) – Function executed as a filter
- `kwargs` – Optional keyword arguments(custom filters)

Результат
None

`channel_post_handler(`*commands=None*, *regexp=None*, *func=None*, *content_types=None*, ***kwargs*`)`

Handles new incoming channel post of any kind - text, photo, sticker, etc. As a parameter to the decorator function, it passes *`telebot.types.Message`* object.

Параметры

- `commands` (`list` of `str`) – Optional list of strings (commands to handle).
- `regexp` (`str`) – Optional regular expression.
- `func` (`function`) – Function executed as a filter
- `content_types` (`list` of `str`) – Supported message content types. Must be a list. Defaults to [„text“].
- `kwargs` – Optional keyword arguments(custom filters)

Результат
None

`chat_join_request_handler`(*func=None*, ***kwargs*)

Handles a request to join the chat has been sent. The bot must have the can_invite_users administrator right in the chat to receive these updates. As a parameter to the decorator function, it passes *telebot.types.ChatJoinRequest* object.

Параметры

- `func` (`function`) – Function executed as a filter
- `kwargs` – Optional keyword arguments(custom filters)

Результат
None

`chat_member_handler`(*func=None*, ***kwargs*)

Handles update in a status of a user in a chat. The bot must be an administrator in the chat and must explicitly specify "chat_member" in the list of allowed_updates to receive these updates. As a parameter to the decorator function, it passes *telebot.types.ChatMemberUpdated* object.

Параметры

- `func` (`function`) – Function executed as a filter
- `kwargs` – Optional keyword arguments(custom filters)

Результат
None

`chosen_inline_handler`(*func*, ***kwargs*)

Handles the result of an inline query that was chosen by a user and sent to their chat partner. Please see our documentation on the feedback collecting for details on how to enable these updates for your bot. As a parameter to the decorator function, it passes *telebot.types.ChosenInlineResult* object.

Параметры

- `func` (`function`) – Function executed as a filter
- `kwargs` – Optional keyword arguments(custom filters)

Результат
None

`clear_reply_handlers`(*message:* Message) → None

Clears all callback functions registered by register_for_reply() and register_for_reply_by_message_id().

Параметры
`message` (*telebot.types.Message*) – The message for which we want to clear reply handlers

Результат
None

`clear_reply_handlers_by_message_id`(*message_id: int*) → None

Clears all callback functions registered by register_for_reply() and register_for_reply_by_message_id().

Параметры
`message_id` (`int`) – The message id for which we want to clear reply handlers

Результат
None

`clear_step_handler`(*message:* Message) → None

Clears all callback functions registered by register_next_step_handler().

Параметры
`message` (*telebot.types.Message*) – The message for which we want to handle new message after that in same chat.

Результат
None

`clear_step_handler_by_chat_id`(*chat_id: Union[int, str]*) → None

Clears all callback functions registered by register_next_step_handler().

Параметры
`chat_id` (`int` or `str`) – The chat for which we want to clear next step handlers

Результат
None

`close`() → bool

Use this method to close the bot instance before moving it from one local server to another. You need to delete the webhook before calling this method to ensure that the bot isn't launched again after server restart. The method will return error 429 in the first 10 minutes after the bot is launched. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#close

Результат
`bool`

`close_forum_topic`(*chat_id: Union[str, int]*, *message_thread_id: int*) → bool

Use this method to close an open topic in a forum supergroup chat. The bot must be an administrator in the chat for this to work and must have the can_manage_topics administrator rights, unless it is the creator of the topic. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#closeforumtopic

Параметры
- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `message_thread_id` (`int`) – Identifier of the topic to close

Результат
On success, True is returned.

Тип результата
`bool`

`copy_message`(*chat_id: Union[int, str]*, *from_chat_id: Union[int, str]*, *message_id: int*, *caption: Optional[str] = None*, *parse_mode: Optional[str] = None*, *caption_entities: Optional[List[MessageEntity]] = None*, *disable_notification: Optional[bool] = None*, *protect_content: Optional[bool] = None*, *reply_to_message_id: Optional[int] = None*, *allow_sending_without_reply: Optional[bool] = None*, *reply_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None*, *timeout: Optional[int] = None*, *message_thread_id: Optional[int] = None*) → *MessageID*

Use this method to copy messages of any kind.

Telegram documentation: https://core.telegram.org/bots/api#copymessage

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `from_chat_id` (`int` or `str`) – Unique identifier for the chat where the original message was sent (or channel username in the format @channelusername)
- `message_id` (`int`) – Message identifier in the chat specified in from_chat_id
- `caption` (`str`) – New caption for media, 0-1024 characters after entities parsing. If not specified, the original caption is kept
- `parse_mode` (`str`) – Mode for parsing entities in the new caption.
- `caption_entities` (Array of *telebot.types.MessageEntity*) – A JSON-serialized list of special entities that appear in the new caption, which can be specified instead of parse_mode
- `disable_notification` (`bool`) – Sends the message silently. Users will receive a notification with no sound.
- `protect_content` (`bool`) – Protects the contents of the sent message from forwarding and saving
- `reply_to_message_id` (`int`) – If the message is a reply, ID of the original message
- `allow_sending_without_reply` (`bool`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- `reply_markup` (*telebot.types.InlineKeyboardMarkup* or *telebot.types.ReplyKeyboardMarkup* or *telebot.types.ReplyKeyboardRemove* or *telebot.types.ForceReply*) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user.
- `timeout` (`int`) – Timeout in seconds for the request.
- `message_thread_id` (`int`) – Identifier of a message thread, in which the message will be sent

Результат

On success, the sent Message is returned.

Тип результата

telebot.types.Message

`create_chat_invite_link`(*chat_id: Union[int, str]*, *name: Optional[str] = None*, *expire_date: Optional[Union[int, datetime]] = None*, *member_limit: Optional[int] = None*, *creates_join_request: Optional[bool] = None*) → *ChatInviteLink*

Use this method to create an additional invite link for a chat. The bot must be an administrator in the chat for this to work and must have the appropriate administrator rights. The link can be revoked using the method revokeChatInviteLink. Returns the new invite link as ChatInviteLink object.

Telegram documentation: https://core.telegram.org/bots/api#createchatinvitelink

Параметры

- `chat_id` (`int` or `str`) – Id: Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `name` (`str`) – Invite link name; 0-32 characters
- `expire_date` (`int` or `datetime`) – Point in time (Unix timestamp) when the link will expire
- `member_limit` (`int`) – Maximum number of users that can be members of the chat simultaneously
- `creates_join_request` (`bool`) – True, if users joining the chat via the link need to be approved by chat administrators. If True, member_limit can't be specified

Результат
Returns the new invite link as ChatInviteLink object.

Тип результата
telebot.types.ChatInviteLink

`create_forum_topic`(*chat_id: int*, *name: str*, *icon_color: Optional[int] = None*, *icon_custom_emoji_id: Optional[str] = None*) → *ForumTopic*

Use this method to create a topic in a forum supergroup chat. The bot must be an administrator in the chat for this to work and must have the can_manage_topics administrator rights. Returns information about the created topic as a ForumTopic object.

Telegram documentation: https://core.telegram.org/bots/api#createforumtopic

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `name` (`str`) – Name of the topic, 1-128 characters
- `icon_color` (`int`) – Color of the topic icon in RGB format. Currently, must be one of 0x6FB9F0, 0xFFD67E, 0xCB86DB, 0x8EEE98, 0xFF93B2, or 0xFB6F5F
- `icon_custom_emoji_id` (`str`) – Custom emoji for the topic icon. Must be an emoji of type "tgs" and must be exactly 1 character long

Результат
On success, information about the created topic is returned as a ForumTopic object.

Тип результата
telebot.types.ForumTopic

`create_invoice_link`(*title: str*, *description: str*, *payload: str*, *provider_token: str*, *currency: str*, *prices: List[LabeledPrice]*, *max_tip_amount: Optional[int] = None*, *suggested_tip_amounts: Optional[List[int]] = None*, *provider_data: Optional[str] = None*, *photo_url: Optional[str] = None*, *photo_size: Optional[int] = None*, *photo_width: Optional[int] = None*, *photo_height: Optional[int] = None*, *need_name: Optional[bool] = None*, *need_phone_number: Optional[bool] = None*, *need_email: Optional[bool] = None*, *need_shipping_address: Optional[bool] = None*, *send_phone_number_to_provider: Optional[bool] = None*, *send_email_to_provider: Optional[bool] = None*, *is_flexible: Optional[bool] = None*) → str

Use this method to create a link for an invoice. Returns the created invoice link as String on success.

Telegram documentation: https://core.telegram.org/bots/api#createinvoicelink

Параметры

- `title` (`str`) – Product name, 1-32 characters
- `description` (`str`) – Product description, 1-255 characters
- `payload` (`str`) – Bot-defined invoice payload, 1-128 bytes. This will not be displayed to the user, use for your internal processes.
- `provider_token` (`str`) – Payments provider token, obtained via @Botfather
- `currency` (`str`) – Three-letter ISO 4217 currency code, see https://core.telegram.org/bots/payments#supported-currencies
- `prices` (`list of types.LabeledPrice`) – Price breakdown, a list of components (e.g. product price, tax, discount, delivery cost, delivery tax, bonus, etc.)
- `max_tip_amount` (`int`) – The maximum accepted amount for tips in the smallest units of the currency
- `suggested_tip_amounts` (`list of int`) – A JSON-serialized array of suggested amounts of tips in the smallest units of the currency. At most 4 suggested tip amounts can be specified. The suggested tip amounts must be positive, passed in a strictly increased order and must not exceed max_tip_amount.
- `provider_data` (`str`) – A JSON-serialized data about the invoice, which will be shared with the payment provider. A detailed description of required fields should be provided by the payment provider.
- `photo_url` (`str`) – URL of the product photo for the invoice. Can be a photo of the goods or a photo of the invoice. People like it better when they see a photo of what they are paying for.
- `photo_size` (`int`) – Photo size in bytes
- `photo_width` (`int`) – Photo width
- `photo_height` (`int`) – Photo height
- `need_name` (`bool`) – Pass True, if you require the user's full name to complete the order
- `need_phone_number` (`bool`) – Pass True, if you require the user's phone number to complete the order
- `need_email` (`bool`) – Pass True, if you require the user's email to complete the order
- `need_shipping_address` (`bool`) – Pass True, if you require the user's shipping address to complete the order
- `send_phone_number_to_provider` (`bool`) – Pass True, if user's phone number should be sent to provider
- `send_email_to_provider` (`bool`) – Pass True, if user's email address should be sent to provider
- `is_flexible` (`bool`) – Pass True, if the final price depends on the shipping method

Результат

Created invoice link as String on success.

Тип результата

`str`

`create_new_sticker_set`(*user_id: int*, *name: str*, *title: str*, *emojis: str*, *png_sticker: Optional[Union[str, Any]] = None*, *tgs_sticker: Optional[Union[str, Any]] = None*, *webm_sticker: Optional[Union[str, Any]] = None*, *contains_masks: Optional[bool] = None*, *sticker_type: Optional[str] = None*, *mask_position: Optional[MaskPosition] = None*) → bool

Use this method to create new sticker set owned by a user. The bot will be able to edit the created sticker set. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#createnewstickerset

Параметры

- `user_id` (`int`) – User identifier of created sticker set owner
- `name` (`str`) – Short name of sticker set, to be used in t.me/addstickers/ URLs (e.g., animals). Can contain only English letters, digits and underscores. Must begin with a letter, can't contain consecutive underscores and must end in «_by_<bot_username>». <bot_username> is case insensitive. 1-64 characters.
- `title` (`str`) – Sticker set title, 1-64 characters
- `emojis` (`str`) – One or more emoji corresponding to the sticker
- `png_sticker` (`str`) – PNG image with the sticker, must be up to 512 kilobytes in size, dimensions must not exceed 512px, and either width or height must be exactly 512px. Pass a file_id as a String to send a file that already exists on the Telegram servers, pass an HTTP URL as a String for Telegram to get a file from the Internet, or upload a new one using multipart/form-data.
- `tgs_sticker` (`str`) – TGS animation with the sticker, uploaded using multipart/form-data.
- `webm_sticker` (`str`) – WebM animation with the sticker, uploaded using multipart/form-data.
- `contains_masks` (`bool`) – Pass True, if a set of mask stickers should be created. Deprecated since Bot API 6.2, use sticker_type instead.
- `sticker_type` (`str`) – Optional, Type of stickers in the set, pass "regular" or "mask". Custom emoji sticker sets can't be created via the Bot API at the moment. By default, a regular sticker set is created.
- `mask_position` (*telebot.types.MaskPosition*) – A JSON-serialized object for position where the mask should be placed on faces

Результат
On success, True is returned.

Тип результата
`bool`

`decline_chat_join_request`(*chat_id: Union[str, int]*, *user_id: Union[int, str]*) → bool

Use this method to decline a chat join request. The bot must be an administrator in the chat for this to work and must have the can_invite_users administrator right. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#declinechatjoinrequest

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target supergroup (in the format @supergroupusername)
- `user_id` (`int` or `str`) – Unique identifier of the target user

Результат
True on success.

Тип результата
`bool`

`delete_chat_photo(`*chat_id: Union[int, str]*`)` → bool

Use this method to delete a chat photo. Photos can't be changed for private chats. The bot must be an administrator in the chat for this to work and must have the appropriate admin rights. Returns True on success. Note: In regular groups (non-supergroups), this method will only work if the 'All Members Are Admins' setting is off in the target group.

Telegram documentation: https://core.telegram.org/bots/api#deletechatphoto

Параметры
`chat_id` (`int` or `str`) – Int or Str: Unique identifier for the target chat or username of the target channel (in the format @channelusername)

Результат
True on success.

Тип результата
`bool`

`delete_chat_sticker_set(`*chat_id: Union[int, str]*`)` → bool

Use this method to delete a group sticker set from a supergroup. The bot must be an administrator in the chat for this to work and must have the appropriate admin rights. Use the field can_set_sticker_set optionally returned in getChat requests to check if the bot can use this method. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#deletechatstickerset

Параметры
`chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target supergroup (in the format @supergroupusername)

Результат
Returns True on success.

Тип результата
`bool`

`delete_forum_topic(`*chat_id: Union[str, int]*, *message_thread_id: int*`)` → bool

Use this method to delete a topic in a forum supergroup chat. The bot must be an administrator in the chat for this to work and must have the can_manage_topics administrator rights, unless it is the creator of the topic. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#deleteforumtopic

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `message_thread_id` (`int`) – Identifier of the topic to delete

Результат
On success, True is returned.

Тип результата
`bool`

`delete_message`(*chat_id: Union[int, str]*, *message_id: int*, *timeout: Optional[int] = None*) → bool

Use this method to delete a message, including service messages, with the following limitations: - A message can only be deleted if it was sent less than 48 hours ago. - A dice message in a private chat can only be deleted if it was sent more than 24 hours ago. - Bots can delete outgoing messages in private chats, groups, and supergroups. - Bots can delete incoming messages in private chats. - Bots granted can_post_messages permissions can delete outgoing messages in channels. - If the bot is an administrator of a group, it can delete any message there. - If the bot has can_delete_messages permission in a supergroup or a channel, it can delete any message there. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#deletemessage

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `message_id` (`int`) – Identifier of the message to delete
- `timeout` (`int`) – Timeout in seconds for the request.

Результат
Returns True on success.

Тип результата
`bool`

`delete_my_commands`(*scope: Optional[BotCommandScope] = None*, *language_code: Optional[str] = None*) → bool

Use this method to delete the list of the bot's commands for the given scope and user language. After deletion, higher level commands will be shown to affected users. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#deletemycommands

Параметры

- `scope` (*`telebot.types.BotCommandScope`*) – The scope of users for which the commands are relevant. Defaults to BotCommandScopeDefault.
- `language_code` (`str`) – A two-letter ISO 639-1 language code. If empty, commands will be applied to all users from the given scope, for whose language there are no dedicated commands

Результат
True on success.

Тип результата
`bool`

`delete_state`(*user_id: int*, *chat_id: Optional[int] = None*) → None

Delete the current state of a user.

Параметры

- `user_id` (`int`) – User's identifier
- `chat_id` (`int`) – Chat's identifier

Результат
None

`delete_sticker_from_set(`*sticker: str*`)` → bool

Use this method to delete a sticker from a set created by the bot. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#deletestickerfromset

Параметры
`sticker` – File identifier of the sticker

Результат
On success, True is returned.

Тип результата
`bool`

`delete_webhook(`*drop_pending_updates: Optional[bool] = None*, *timeout: Optional[int] = None*`)` → bool

Use this method to remove webhook integration if you decide to switch back to getUpdates. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#deletewebhook

Параметры

- `drop_pending_updates` – Pass True to drop all pending updates, defaults to None
- `timeout` (`int`, optional) – Request connection timeout, defaults to None

Результат
Returns True on success.

Тип результата
`bool`

`disable_save_next_step_handlers()`

Disable saving next step handlers (by default saving disable)

This function is left to keep backward compatibility whose purpose was to disable file saving capability for handlers. For the same purpose, MemoryHandlerBackend is reassigned as a new next_step_backend backend instead of FileHandlerBackend.

`disable_save_reply_handlers()`

Disable saving next step handlers (by default saving disable)

This function is left to keep backward compatibility whose purpose was to disable file saving capability for handlers. For the same purpose, MemoryHandlerBackend is reassigned as a new reply_backend backend instead of FileHandlerBackend.

`download_file(`*file_path: str*`)` → bytes

Downloads file.

Параметры
`file_path` (*`str`*) – Path where the file should be downloaded.

Результат
bytes

Тип результата
`bytes`

`edit_chat_invite_link(`*chat_id: Union[int, str]*, *invite_link: Optional[str] = None*, *name: Optional[str] = None*, *expire_date: Optional[Union[int, datetime]] = None*, *member_limit: Optional[int] = None*, *creates_join_request: Optional[bool] = None*`)` → *ChatInviteLink*

Use this method to edit a non-primary invite link created by the bot. The bot must be an administrator in the chat for this to work and must have the appropriate admin rights.

Telegram documentation: https://core.telegram.org/bots/api#editchatinvitelink

Параметры

- `chat_id` (`int` or `str`) – Id: Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `name` (`str`) – Invite link name; 0-32 characters
- `invite_link` (`str`) – The invite link to edit
- `expire_date` (`int` or `datetime`) – Point in time (Unix timestamp) when the link will expire
- `member_limit` (`int`) – Maximum number of users that can be members of the chat simultaneously
- `creates_join_request` (`bool`) – True, if users joining the chat via the link need to be approved by chat administrators. If True, member_limit can't be specified

Результат
Returns the new invite link as ChatInviteLink object.

Тип результата
telebot.types.ChatInviteLink

edit_forum_topic(*chat_id: Union[int, str]*, *message_thread_id: int*, *name: str*, *icon_custom_emoji_id: str*) → bool

Use this method to edit name and icon of a topic in a forum supergroup chat. The bot must be an administrator in the chat for this to work and must have can_manage_topics administrator rights, unless it is the creator of the topic. Returns True on success.

Telegram Documentation: https://core.telegram.org/bots/api#editforumtopic

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `message_thread_id` (`int`) – Identifier of the topic to edit
- `name` (`str`) – New name of the topic, 1-128 characters
- `icon_custom_emoji_id` (`str`) – New custom emoji for the topic icon. Must be an emoji of type "tgs" and must be exactly 1 character long

Результат
On success, True is returned.

Тип результата
`bool`

edit_message_caption(*caption: str*, *chat_id: Optional[Union[int, str]] = None*, *message_id: Optional[int] = None*, *inline_message_id: Optional[str] = None*, *parse_mode: Optional[str] = None*, *caption_entities: Optional[List[MessageEntity]] = None*, *reply_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None*) → Union[*Message*, bool]

Use this method to edit captions of messages.

Telegram documentation: https://core.telegram.org/bots/api#editmessagecaption

Параметры

- `caption` (`str`) – New caption of the message
- `chat_id` (`int` | `str`) – Required if inline_message_id is not specified. Unique identifier for the target chat or username of the target channel
- `message_id` (`int`) – Required if inline_message_id is not specified.
- `inline_message_id` (`str`) – Required if inline_message_id is not specified. Identifier of the inline message.
- `parse_mode` (`str`) – New caption of the message, 0-1024 characters after entities parsing
- `caption_entities` (`list` of *types.MessageEntity*) – A JSON-serialized array of objects that describe how the caption should be parsed.
- `reply_markup` (`InlineKeyboardMarkup`) – A JSON-serialized object for an inline keyboard.

Результат

On success, if edited message is sent by the bot, the edited Message is returned, otherwise True is returned.

Тип результата

types.Message | `bool`

`edit_message_live_location`(*latitude: float*, *longitude: float*, *chat_id: Optional[Union[int, str]] = None*, *message_id: Optional[int] = None*, *inline_message_id: Optional[str] = None*, *reply_markup: Optional[Union[*InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply*]] = None*, *timeout: Optional[int] = None*, *horizontal_accuracy: Optional[float] = None*, *heading: Optional[int] = None*, *proximity_alert_radius: Optional[int] = None*) → *Message*

Use this method to edit live location messages. A location can be edited until its live_period expires or editing is explicitly

disabled by a call to stopMessageLiveLocation. On success, if the edited message is not an inline message, the edited Message is returned, otherwise True is returned.

Telegram documentation: https://core.telegram.org/bots/api#editmessagelivelocation

Параметры

- `latitude` (`float`) – Latitude of new location
- `longitude` (`float`) – Longitude of new location
- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `message_id` (`int`) – Required if inline_message_id is not specified. Identifier of the message to edit
- `reply_markup` (*telebot.types.InlineKeyboardMarkup* or *telebot.types.ReplyKeyboardMarkup* or *telebot.types.ReplyKeyboardRemove* or *telebot.types.ForceReply*) – A JSON-serialized object for a new inline keyboard.

- `timeout` (`int`) – Timeout in seconds for the request.
- `inline_message_id` (`str`) – Required if chat_id and message_id are not specified. Identifier of the inline message
- `horizontal_accuracy` (`float`) – The radius of uncertainty for the location, measured in meters; 0-1500
- `heading` (`int`) – Direction in which the user is moving, in degrees. Must be between 1 and 360 if specified.
- `proximity_alert_radius` (`int`) – The maximum distance for proximity alerts about approaching another chat member, in meters. Must be between 1 and 100000 if specified.

Результат
On success, if the edited message is not an inline message, the edited Message is returned, otherwise True is returned.

Тип результата
telebot.types.Message or bool

`edit_message_media`(*media: Any*, *chat_id: Optional[Union[int, str]] = None*, *message_id: Optional[int] = None*, *inline_message_id: Optional[str] = None*, *reply_markup: Optional[Union[*InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply*]] = None*) → Union[*Message*, bool]

Use this method to edit animation, audio, document, photo, or video messages. If a message is a part of a message album, then it can be edited only to a photo or a video. Otherwise, message type can be changed arbitrarily. When inline message is edited, new file can't be uploaded. Use previously uploaded file via its file_id or specify a URL.

Telegram documentation: https://core.telegram.org/bots/api#editmessagemedia

Параметры

- `media` (`InputMedia`) – A JSON-serialized object for a new media content of the message
- `chat_id` (`int` or `str`) – Required if inline_message_id is not specified. Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `message_id` (`int`) – Required if inline_message_id is not specified. Identifier of the sent message
- `inline_message_id` (`str`) – Required if chat_id and message_id are not specified. Identifier of the inline message
- `reply_markup` (*telebot.types.InlineKeyboardMarkup* or `ReplyKeyboardMarkup` or `ReplyKeyboardRemove` or `ForceReply`) – A JSON-serialized object for an inline keyboard.

Результат
On success, if edited message is sent by the bot, the edited Message is returned, otherwise True is returned.

Тип результата
types.Message or `bool`

edit_message_reply_markup(*chat_id: Optional[Union[int, str]] = None*, *message_id: Optional[int] = None*, *inline_message_id: Optional[str] = None*, *reply_markup: Optional[Union[*InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply*]] = None*) → Union[*Message*, bool]

Use this method to edit only the reply markup of messages.

Telegram documentation: https://core.telegram.org/bots/api#editmessagereplymarkup

Параметры

- `chat_id` (`int` or `str`) – Required if inline_message_id is not specified. Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `message_id` (`int`) – Required if inline_message_id is not specified. Identifier of the sent message
- `inline_message_id` (`str`) – Required if chat_id and message_id are not specified. Identifier of the inline message
- `reply_markup` (`InlineKeyboardMarkup` or `ReplyKeyboardMarkup` or `ReplyKeyboardRemove` or `ForceReply`) – A JSON-serialized object for an inline keyboard.

Результат

On success, if edited message is sent by the bot, the edited Message is returned, otherwise True is returned.

Тип результата

types.Message or `bool`

edit_message_text(*text: str*, *chat_id: Optional[Union[int, str]] = None*, *message_id: Optional[int] = None*, *inline_message_id: Optional[str] = None*, *parse_mode: Optional[str] = None*, *entities: Optional[List[*MessageEntity*]] = None*, *disable_web_page_preview: Optional[bool] = None*, *reply_markup: Optional[Union[*InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply*]] = None*) → Union[*Message*, bool]

Use this method to edit text and game messages.

Telegram documentation: https://core.telegram.org/bots/api#editmessagetext

Параметры

- `text` (`str`) – New text of the message, 1-4096 characters after entities parsing
- `chat_id` (`int` or `str`) – Required if inline_message_id is not specified. Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `message_id` (`int`) – Required if inline_message_id is not specified. Identifier of the sent message
- `inline_message_id` (`str`) – Required if chat_id and message_id are not specified. Identifier of the inline message
- `parse_mode` (`str`) – Mode for parsing entities in the message text.
- `entities` (List of *telebot.types.MessageEntity*) – List of special entities that appear in the message text, which can be specified instead of parse_mode

- `disable_web_page_preview` (`bool`) – Disables link previews for links in this message
- `reply_markup` (`InlineKeyboardMarkup`) – A JSON-serialized object for an inline keyboard.

Результат
On success, if edited message is sent by the bot, the edited Message is returned, otherwise True is returned.

Тип результата
types.Message or `bool`

`edited_channel_post_handler(`*commands=None*, *regexp=None*, *func=None*, *content_types=None*, ***kwargs*`)`

Handles new version of a channel post that is known to the bot and was edited. As a parameter to the decorator function, it passes *telebot.types.Message* object.

Параметры

- `commands` (`list` of `str`) – Optional list of strings (commands to handle).
- `regexp` (`str`) – Optional regular expression.
- `func` (`function`) – Function executed as a filter
- `content_types` (`list` of `str`) – Supported message content types. Must be a list. Defaults to [„text“].
- `kwargs` – Optional keyword arguments(custom filters)

Результат

`edited_message_handler(`*commands=None*, *regexp=None*, *func=None*, *content_types=None*, *chat_types=None*, ***kwargs*`)`

Handles new version of a message that is known to the bot and was edited. As a parameter to the decorator function, it passes *telebot.types.Message* object.

Параметры

- `commands` (`list` of `str`) – Optional list of strings (commands to handle).
- `regexp` (`str`) – Optional regular expression.
- `func` (`function`) – Function executed as a filter
- `content_types` (`list` of `str`) – Supported message content types. Must be a list. Defaults to [„text“].
- `chat_types` (`list` of `str`) – list of chat types
- `kwargs` – Optional keyword arguments(custom filters)

Результат
None

`enable_save_next_step_handlers(`*delay: Optional[int] = 120*, *filename: Optional[str] = './.handler-saves/step.save'*`)`

Enable saving next step handlers (by default saving disabled)

This function explicitly assigns FileHandlerBackend (instead of Saver) just to keep backward compatibility whose purpose was to enable file saving capability for handlers. And the same implementation is now available with FileHandlerBackend

Параметры

- `delay` (`int`, optional) – Delay between changes in handlers and saving, defaults to 120
- `filename` (`str`, optional) – Filename of save file, defaults to «./.handler-saves/step.save»

Результат
None

`enable_save_reply_handlers`(*delay=120*, *filename='./.handler-saves/reply.save'*)

Enable saving reply handlers (by default saving disable)

This function explicitly assigns FileHandlerBackend (instead of Saver) just to keep backward compatibility whose purpose was to enable file saving capability for handlers. And the same implementation is now available with FileHandlerBackend

Параметры

- `delay` (`int`, optional) – Delay between changes in handlers and saving, defaults to 120
- `filename` (`str`, optional) – Filename of save file, defaults to «./.handler-saves/reply.save»

`enable_saving_states`(*filename: Optional[str] = './.state-save/states.pkl'*)

Enable saving states (by default saving disabled)

Примечание: It is recommended to pass a `StatePickleStorage` instance as state_storage to TeleBot class.

Параметры
`filename` (`str`, optional) – Filename of saving file, defaults to «./.state-save/states.pkl»

`export_chat_invite_link`(*chat_id: Union[int, str]*) → str

Use this method to export an invite link to a supergroup or a channel. The bot must be an administrator in the chat for this to work and must have the appropriate admin rights.

Telegram documentation: https://core.telegram.org/bots/api#exportchatinvitelink

Параметры
`chat_id` (`int` or `str`) – Id: Unique identifier for the target chat or username of the target channel (in the format @channelusername)

Результат
exported invite link as String on success.

Тип результата
`str`

`forward_message`(*chat_id: Union[int, str]*, *from_chat_id: Union[int, str]*, *message_id: int*, *disable_notification: Optional[bool] = None*, *protect_content: Optional[bool] = None*, *timeout: Optional[int] = None*, *message_thread_id: Optional[int] = None*) → *Message*

Use this method to forward messages of any kind.

Telegram documentation: https://core.telegram.org/bots/api#forwardmessage

Параметры

- `disable_notification` (`bool`) – Sends the message silently. Users will receive a notification with no sound
- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `from_chat_id` (`int` or `str`) – Unique identifier for the chat where the original message was sent (or channel username in the format @channelusername)
- `message_id` (`int`) – Message identifier in the chat specified in from_chat_id
- `protect_content` (`bool`) – Protects the contents of the forwarded message from forwarding and saving
- `timeout` (`int`) – Timeout in seconds for the request.
- `message_thread_id` (`int`) – Identifier of a message thread, in which the message will be sent

Результат
On success, the sent Message is returned.

Тип результата
telebot.types.Message

`get_chat`(*chat_id: Union[int, str]*) → *Chat*

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.). Returns a Chat object on success.

Telegram documentation: https://core.telegram.org/bots/api#getchat

Параметры
`chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target supergroup or channel (in the format @channelusername)

Результат
Chat information

Тип результата
telebot.types.Chat

`get_chat_administrators`(*chat_id: Union[int, str]*) → List[*ChatMember*]

Use this method to get a list of administrators in a chat. On success, returns an Array of ChatMember objects that contains information about all chat administrators except other bots.

Telegram documentation: https://core.telegram.org/bots/api#getchatadministrators

Параметры
`chat_id` – Unique identifier for the target chat or username of the target supergroup or channel (in the format @channelusername)

Результат
List made of ChatMember objects.

Тип результата
`list` of *telebot.types.ChatMember*

`get_chat_member`(*chat_id: Union[int, str]*, *user_id: int*) → *ChatMember*

Use this method to get information about a member of a chat. Returns a ChatMember object on success.

Telegram documentation: https://core.telegram.org/bots/api#getchatmember

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target supergroup (in the format @supergroupusername)
- `user_id` (`int`) – Unique identifier of the target user

Результат
Returns ChatMember object on success.

Тип результата
telebot.types.ChatMember

`get_chat_member_count`(*chat_id: Union[int, str]*) → int

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

Telegram documentation: https://core.telegram.org/bots/api#getchatmembercount

Параметры
`chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target supergroup or channel (in the format @channelusername)

Результат
Number of members in the chat.

Тип результата
`int`

`get_chat_members_count`(***kwargs*)

`get_chat_menu_button`(*chat_id: Optional[Union[int, str]] = None*) → *MenuButton*

Use this method to get the current value of the bot's menu button in a private chat, or the default menu button. Returns MenuButton on success.

Telegram Documentation: https://core.telegram.org/bots/api#getchatmenubutton

Параметры
`chat_id` (`int` or `str`) – Unique identifier for the target private chat. If not specified, default bot's menu button will be returned.

Результат
types.MenuButton

Тип результата
telebot.types.MenuButton

`get_custom_emoji_stickers`(*custom_emoji_ids: List[str]*) → List[*Sticker*]

Use this method to get information about custom emoji stickers by their identifiers. Returns an Array of Sticker objects.

Параметры
`custom_emoji_ids` (`list` of `str`) – List of custom emoji identifiers. At most 200 custom emoji identifiers can be specified.

Результат
Returns an Array of Sticker objects.

Тип результата
`list` of *telebot.types.Sticker*

get_file(*file_id: Optional[str]*) → *File*

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 20MB in size. On success, a File object is returned. It is guaranteed that the link will be valid for at least 1 hour. When the link expires, a new one can be requested by calling get_file again.

Telegram documentation: https://core.telegram.org/bots/api#getfile

Параметры
file_id (str) – File identifier

Результат
telebot.types.File

get_file_url(*file_id: Optional[str]*) → str

Get a valid URL for downloading a file.

Параметры
file_id (str) – File identifier to get download URL for.

Результат
URL for downloading the file.

Тип результата
str

get_forum_topic_icon_stickers() → List[*Sticker*]

Use this method to get custom emoji stickers, which can be used as a forum topic icon by any user. Requires no parameters. Returns an Array of Sticker objects.

Telegram documentation: https://core.telegram.org/bots/api#getforumtopiciconstickers

Результат
On success, a list of StickerSet objects is returned.

Тип результата
List[*telebot.types.StickerSet*]

get_game_high_scores(*user_id: int*, *chat_id: Optional[Union[int, str]] = None*, *message_id: Optional[int] = None*, *inline_message_id: Optional[str] = None*) → List[*GameHighScore*]

Use this method to get data for high score tables. Will return the score of the specified user and several of their neighbors in a game. On success, returns an Array of GameHighScore objects.

This method will currently return scores for the target user, plus two of their closest neighbors on each side. Will also return the top three users if the user and their neighbors are not among them. Please note that this behavior is subject to change.

Telegram documentation: https://core.telegram.org/bots/api#getgamehighscores

Параметры

- user_id (int) – User identifier
- chat_id (int or str) – Required if inline_message_id is not specified. Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- message_id (int) – Required if inline_message_id is not specified. Identifier of the sent message
- inline_message_id (str) – Required if chat_id and message_id are not specified. Identifier of the inline message

Результат
On success, returns an Array of GameHighScore objects.

Тип результата
List[*types.GameHighScore*]

`get_me()` → *User*

A simple method for testing your bot's authentication token. Requires no parameters. Returns basic information about the bot in form of a User object.

Telegram documentation: https://core.telegram.org/bots/api#getme

`get_my_commands`(*scope: Optional[BotCommandScope] = None*, *language_code: Optional[str] = None*) → List[*BotCommand*]

Use this method to get the current list of the bot's commands. Returns List of BotCommand on success.

Telegram documentation: https://core.telegram.org/bots/api#getmycommands

Параметры

- `scope` (*telebot.types.BotCommandScope*) – The scope of users for which the commands are relevant. Defaults to BotCommandScopeDefault.
- `language_code` (`str`) – A two-letter ISO 639-1 language code. If empty, commands will be applied to all users from the given scope, for whose language there are no dedicated commands

Результат
List of BotCommand on success.

Тип результата
`list` of *telebot.types.BotCommand*

`get_my_default_administrator_rights`(*for_channels: Optional[bool] = None*) → *ChatAdministratorRights*

Use this method to get the current default administrator rights of the bot. Returns ChatAdministratorRights on success.

Telegram documentation: https://core.telegram.org/bots/api#getmydefaultadministratorrights

Параметры
`for_channels` (`bool`) – Pass True to get the default administrator rights of the bot in channels. Otherwise, the default administrator rights of the bot for groups and supergroups will be returned.

Результат
Returns ChatAdministratorRights on success.

Тип результата
telebot.types.ChatAdministratorRights

`get_state`(*user_id: int*, *chat_id: Optional[int] = None*) → Optional[Union[int, str, *State*]]

Gets current state of a user. Not recommended to use this method. But it is ok for debugging.

Параметры

- `user_id` (`int`) – User's identifier
- `chat_id` (`int`) – Chat's identifier

Результат
state of a user

Тип результата
int or str or telebot.types.State

get_sticker_set(*name: str*) → *StickerSet*

Use this method to get a sticker set. On success, a StickerSet object is returned.

Telegram documentation: https://core.telegram.org/bots/api#getstickerset

Параметры
name (str) – Sticker set name

Результат
On success, a StickerSet object is returned.

Тип результата
telebot.types.StickerSet

get_updates(*offset: Optional[int] = None*, *limit: Optional[int] = None*, *timeout: Optional[int] = 20*, *allowed_updates: Optional[List[str]] = None*, *long_polling_timeout: int = 20*) → List[*Update*]

Use this method to receive incoming updates using long polling (wiki). An Array of Update objects is returned.

Telegram documentation: https://core.telegram.org/bots/api#getupdates

Параметры

- offset (int, optional) – Identifier of the first update to be returned. Must be greater by one than the highest among the identifiers of previously received updates. By default, updates starting with the earliest unconfirmed update are returned. An update is considered confirmed as soon as getUpdates is called with an offset higher than its update_id. The negative offset can be specified to retrieve updates starting from -offset update from the end of the updates queue. All previous updates will forgotten.
- limit (int, optional) – Limits the number of updates to be retrieved. Values between 1-100 are accepted. Defaults to 100.
- timeout (int, optional) – Request connection timeout
- allowed_updates (list, optional) – Array of string. List the types of updates you want your bot to receive.
- long_polling_timeout (int, optional) – Timeout in seconds for long polling.

Результат
An Array of Update objects is returned.

Тип результата
list of *telebot.types.Update*

get_user_profile_photos(*user_id: int*, *offset: Optional[int] = None*, *limit: Optional[int] = None*) → *UserProfilePhotos*

Use this method to get a list of profile pictures for a user. Returns a *telebot.types.UserProfilePhotos* object.

Telegram documentation: https://core.telegram.org/bots/api#getuserprofilephotos

Параметры

- user_id (int) – Unique identifier of the target user

- `offset` (`int`) – Sequential number of the first photo to be returned. By default, all photos are returned.
- `limit` (`int`) – Limits the number of photos to be retrieved. Values between 1-100 are accepted. Defaults to 100.

Результат
UserProfilePhotos

Тип результата
telebot.types.UserProfilePhotos

`get_webhook_info`(*timeout: Optional[int] = None*) → *WebhookInfo*

Use this method to get current webhook status. Requires no parameters. On success, returns a WebhookInfo object. If the bot is using getUpdates, will return an object with the url field empty.

Telegram documentation: https://core.telegram.org/bots/api#getwebhookinfo

Параметры
`timeout` (`int`, optional) – Request connection timeout

Результат
On success, returns a WebhookInfo object.

Тип результата
telebot.types.WebhookInfo

`infinity_polling`(*timeout: Optional[int] = 20*, *skip_pending: Optional[bool] = False*, *long_polling_timeout: Optional[int] = 20*, *logger_level: Optional[int] = 40*, *allowed_updates: Optional[List[str]] = None*, *restart_on_change: Optional[bool] = False*, *path_to_watch: Optional[str] = None*, **args*, ***kwargs*)

Wrap polling with infinite loop and exception handling to avoid bot stops polling.

Примечание: Install watchdog and psutil before using restart_on_change option.

Параметры

- `timeout` (`int`) – Request connection timeout.
- `long_polling_timeout` (`int`) – Timeout in seconds for long polling (see API docs)
- `skip_pending` (`bool`) – skip old updates
- `logger_level` (`int.`) – Custom (different from logger itself) logging level for infinity_polling logging. Use logger levels from logging as a value. None/NOTSET = no error logging
- `allowed_updates` (`list` of `str`) – A list of the update types you want your bot to receive. For example, specify [“message”, “edited_channel_post”, “callback_query”] to only receive updates of these types. See util.update_types for a complete list of available update types. Specify an empty list to receive all update types except chat_member (default). If not specified, the previous setting will be used. Please note that this parameter doesn’t affect updates created before the call to the get_updates, so unwanted updates may be received for a short period of time.
- `restart_on_change` (`bool`) – Restart a file on file(s) change. Defaults to False
- `path_to_watch` (`str`) – Path to watch for changes. Defaults to current directory

Результат

`inline_handler`(*func*, ***kwargs*)

Handles new incoming inline query. As a parameter to the decorator function, it passes *telebot.types.InlineQuery* object.

Параметры

- `func` (`function`) – Function executed as a filter
- `kwargs` – Optional keyword arguments(custom filters)

Результат
None

`kick_chat_member`(***kwargs*)

`leave_chat`(*chat_id: Union[int, str]*) → bool

Use this method for your bot to leave a group, supergroup or channel. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#leavechat

Параметры
`chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target supergroup or channel (in the format @channelusername)

Результат
`bool`

`load_next_step_handlers`(*filename='./.handler-saves/step.save'*, *del_file_after_loading=True*)

Load next step handlers from save file

This function is left to keep backward compatibility whose purpose was to load handlers from file with the help of FileHandlerBackend and is only recommended to use if next_step_backend was assigned as FileHandlerBackend before entering this function

Параметры

- `filename` (`str`, optional) – Filename of the file where handlers was saved, defaults to «./.handler-saves/step.save»
- `del_file_after_loading` (`bool`, optional) – If True is passed, after the loading file will be deleted, defaults to True

`load_reply_handlers`(*filename='./.handler-saves/reply.save'*, *del_file_after_loading=True*)

Load reply handlers from save file

This function is left to keep backward compatibility whose purpose was to load handlers from file with the help of FileHandlerBackend and is only recommended to use if reply_backend was assigned as FileHandlerBackend before entering this function

Параметры

- `filename` (`str`, optional) – Filename of the file where handlers was saved, defaults to «./.handler-saves/reply.save»
- `del_file_after_loading` (`bool`, optional) – If True is passed, after the loading file will be deleted, defaults to True, defaults to True

`log_out`() → bool

Use this method to log out from the cloud Bot API server before launching the bot locally. You MUST log out the bot before running it locally, otherwise there is no guarantee that the bot will receive updates. After a successful call, you can immediately log in on a local server, but will not be able to log in back to the cloud Bot API server for 10 minutes. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#logout

Результат
True on success.

Тип результата
`bool`

`message_handler`(*commands: Optional[List[str]] = None, regexp: Optional[str] = None, func: Optional[Callable] = None, content_types: Optional[List[str]] = None, chat_types: Optional[List[str]] = None, **kwargs*)

Handles New incoming message of any kind - text, photo, sticker, etc. As a parameter to the decorator function, it passes *telebot.types.Message* object. All message handlers are tested in the order they were added.

Example:

Список 6: Usage of message_handler

```
bot = TeleBot('TOKEN')

# Handles all messages which text matches regexp.
@bot.message_handler(regexp='someregexp')
def command_help(message):
    bot.send_message(message.chat.id, 'Did someone call for help?')

# Handles messages in private chat
@bot.message_handler(chat_types=['private']) # You can add more chat types
def command_help(message):
    bot.send_message(message.chat.id, 'Private chat detected, sir!')

# Handle all sent documents of type 'text/plain'.
@bot.message_handler(func=lambda message: message.document.mime_type == 'text/
↪plain',
    content_types=['document'])
def command_handle_document(message):
    bot.send_message(message.chat.id, 'Document received, sir!')

# Handle all other messages.
@bot.message_handler(func=lambda message: True, content_types=['audio', 'photo',
↪ 'voice', 'video', 'document',
    'text', 'location', 'contact', 'sticker'])
def default_command(message):
    bot.send_message(message.chat.id, "This is the default command handler.")
```

Параметры

- `commands` (`list` of `str`) – Optional list of strings (commands to handle).
- `regexp` (`str`) – Optional regular expression.
- `func` (`lambda`) – Optional lambda function. The lambda receives the message to test as the first parameter. It must return True if the command should handle the message.
- `content_types` (`list` of `str`) – Supported message content types. Must be a list. Defaults to [„text“].

- `chat_types` (`list` of `str`) – list of chat types
- `kwargs` – Optional keyword arguments(custom filters)

Результат
decorated function

`middleware_handler`(*update_types: Optional[List[str]] = None*)

Function-based middleware handler decorator.

This decorator can be used to decorate functions that must be handled as middlewares before entering any other message handlers But, be careful and check type of the update inside the handler if more than one update_type is given

Example:

Список 7: Usage of middleware_handler

```
bot = TeleBot('TOKEN')

# Print post message text before entering to any post_channel handlers
@bot.middleware_handler(update_types=['channel_post', 'edited_channel_post'])
def print_channel_post_text(bot_instance, channel_post):
    print(channel_post.text)

# Print update id before entering to any handlers
@bot.middleware_handler()
def print_channel_post_text(bot_instance, update):
    print(update.update_id)
```

Параметры
`update_types` (`list` of `str`) – Optional list of update types that can be passed into the middleware handler.

Результат
function

`my_chat_member_handler`(*func=None*, ***kwargs*)

Handles update in a status of a bot. For private chats, this update is received only when the bot is blocked or unblocked by the user. As a parameter to the decorator function, it passes *telebot.types.ChatMemberUpdated* object.

Параметры

- `func` (`function`) – Function executed as a filter
- `kwargs` – Optional keyword arguments(custom filters)

Результат
None

`pin_chat_message`(*chat_id: Union[int, str]*, *message_id: int*, *disable_notification: Optional[bool] = False*) → bool

Use this method to pin a message in a supergroup. The bot must be an administrator in the chat for this to work and must have the appropriate admin rights. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#pinchatmessage

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `message_id` (`int`) – Identifier of a message to pin
- `disable_notification` (`bool`) – Pass True, if it is not necessary to send a notification to all group members about the new pinned message

Результат
True on success.

Тип результата
`bool`

`poll_answer_handler(func=None, **kwargs)`

Handles change of user's answer in a non-anonymous poll(when user changes the vote). Bots receive new votes only in polls that were sent by the bot itself. As a parameter to the decorator function, it passes *telebot.types.PollAnswer* object.

Параметры

- `func` (`function`) – Function executed as a filter
- `kwargs` – Optional keyword arguments(custom filters)

Результат
None

`poll_handler(func, **kwargs)`

Handles new state of a poll. Bots receive only updates about stopped polls and polls, which are sent by the bot As a parameter to the decorator function, it passes *telebot.types.Poll* object.

Параметры

- `func` (`function`) – Function executed as a filter
- `kwargs` – Optional keyword arguments(custom filters)

Результат
None

`polling(non_stop: Optional[bool] = False, skip_pending: Optional[bool] = False, interval: Optional[int] = 0, timeout: Optional[int] = 20, long_polling_timeout: Optional[int] = 20, logger_level: Optional[int] = 40, allowed_updates: Optional[List[str]] = None, none_stop: Optional[bool] = None, restart_on_change: Optional[bool] = False, path_to_watch: Optional[str] = None)`

This function creates a new Thread that calls an internal __retrieve_updates function. This allows the bot to retrieve Updates automatically and notify listeners and message handlers accordingly.

Warning: Do not call this function more than once!

Always gets updates.

Не рекомендуется, начиная с версии 4.1.1: Use *infinity_polling()* instead.

Примечание: Install watchdog and psutil before using restart_on_change option.

Параметры

- `interval` (`int`) – Delay between two update retrivals

- `non_stop` (`bool`) – Do not stop polling when an ApiException occurs.
- `timeout` (`int`) – Request connection timeout
- `skip_pending` (`bool`) – skip old updates
- `long_polling_timeout` (`int`) – Timeout in seconds for long polling (see API docs)
- `logger_level` (`int`) – Custom (different from logger itself) logging level for infinity_polling logging. Use logger levels from logging as a value. None/NOTSET = no error logging
- `allowed_updates` (`list` of `str`) – A list of the update types you want your bot to receive. For example, specify [“message”, “edited_channel_post”, “callback_query”] to only receive updates of these types. See util.update_types for a complete list of available update types. Specify an empty list to receive all update types except chat_member (default). If not specified, the previous setting will be used.

  Please note that this parameter doesn't affect updates created before the call to the get_updates, so unwanted updates may be received for a short period of time.
- `none_stop` (`bool`) – Deprecated, use non_stop. Old typo, kept for backward compatibility.
- `restart_on_change` (`bool`) – Restart a file on file(s) change. Defaults to False
- `path_to_watch` (`str`) – Path to watch for changes. Defaults to None

Результат

`pre_checkout_query_handler`(*func*, ***kwargs*)

New incoming pre-checkout query. Contains full information about checkout. As a parameter to the decorator function, it passes *telebot.types.PreCheckoutQuery* object.

Параметры

- `func` (`function`) – Function executed as a filter
- `kwargs` – Optional keyword arguments(custom filters)

Результат

None

`process_new_updates`(*updates: List[*Update*]*)

Processes new updates. Just pass list of subclasses of Update to this method.

Параметры

`updates` (`list` of *telebot.types.Update*) – List of *telebot.types.Update* objects.

Return None

`promote_chat_member`(*chat_id: Union[int, str]*, *user_id: int*, *can_change_info: Optional[bool] = None*, *can_post_messages: Optional[bool] = None*, *can_edit_messages: Optional[bool] = None*, *can_delete_messages: Optional[bool] = None*, *can_invite_users: Optional[bool] = None*, *can_restrict_members: Optional[bool] = None*, *can_pin_messages: Optional[bool] = None*, *can_promote_members: Optional[bool] = None*, *is_anonymous: Optional[bool] = None*, *can_manage_chat: Optional[bool] = None*, *can_manage_video_chats: Optional[bool] = None*, *can_manage_voice_chats: Optional[bool] = None*, *can_manage_topics: Optional[bool] = None*) → bool

Use this method to promote or demote a user in a supergroup or a channel. The bot must be 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.

Telegram documentation: https://core.telegram.org/bots/api#promotechatmember

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `user_id` (`int`) – Unique identifier of the target user
- `can_change_info` (`bool`) – Pass True, if the administrator can change chat title, photo and other settings
- `can_post_messages` (`bool`) – Pass True, if the administrator can create channel posts, channels only
- `can_edit_messages` (`bool`) – Pass True, if the administrator can edit messages of other users, 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 administrator 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 administrator can pin messages, supergroups 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 him)
- `is_anonymous` (`bool`) – Pass True, if the administrator's presence in the chat is hidden
- `can_manage_chat` (`bool`) – Pass True, if the administrator can access the chat event log, chat statistics, message statistics in channels, see channel members, see anonymous administrators in supergroups and ignore slow mode. Implied by any other administrator privilege
- `can_manage_video_chats` (`bool`) – Pass True, if the administrator can manage voice chats For now, bots can use this privilege only for passing to other administrators.
- `can_manage_voice_chats` (`bool`) – Deprecated, use can_manage_video_chats.
- `can_manage_topics` (`bool`) – Pass True if the user is allowed to create, rename, close, and reopen forum topics, supergroups only

Результат

True on success.

Тип результата

`bool`

`register_callback_query_handler`(*callback: Callable*, *func: Callable*, *pass_ bot: Optional[bool] = False*, ***kwargs*)

Registers callback query handler.

Параметры

- `callback` (`function`) – function to be called
- `func` (`function`) – Function executed as a filter
- `pass_bot` (`bool`) – True if you need to pass TeleBot instance to handler(useful for separating handlers into different files)
- `kwargs` – Optional keyword arguments(custom filters)

Результат
None

`register_channel_post_handler`(*callback: Callable*, *content_ types: Optional[List[str]] = None*, *commands: Optional[List[str]] = None*, *regexp: Optional[str] = None*, *func: Optional[Callable] = None*, *pass_ bot: Optional[bool] = False*, ***kwargs*)

Registers channel post message handler.

Параметры

- `callback` (`function`) – function to be called
- `content_types` (`list` of `str`) – Supported message content types. Must be a list. Defaults to [„text“].
- `commands` (`list` of `str`) – list of commands
- `regexp` (`str`) – Regular expression
- `func` (`function`) – Function executed as a filter
- `pass_bot` (`bool`) – True if you need to pass TeleBot instance to handler(useful for separating handlers into different files)
- `kwargs` – Optional keyword arguments(custom filters)

Результат
None

`register_chat_join_request_handler`(*callback: Callable*, *func: Optional[Callable] = None*, *pass_ bot: Optional[bool] = False*, ***kwargs*)

Registers chat join request handler.

Параметры

- `callback` (`function`) – function to be called
- `func` (`function`) – Function executed as a filter
- `pass_bot` (`bool`) – True if you need to pass TeleBot instance to handler(useful for separating handlers into different files)
- `kwargs` – Optional keyword arguments(custom filters)

Результат
None

`register_chat_member_handler`(*callback: Callable*, *func: Optional[Callable] = None*, *pass_ bot: Optional[bool] = False*, ***kwargs*)

Registers chat member handler.

Параметры

- `callback` (`function`˙) – function to be called
- `func` (`function`) – Function executed as a filter
- `pass_bot` (`bool`) – True if you need to pass TeleBot instance to handler(useful for separating handlers into different files)
- `kwargs` – Optional keyword arguments(custom filters)

:return:None

`register_chosen_inline_handler`(*callback: Callable*, *func: Callable*, *pass_ bot: Optional[bool] = False*, ***kwargs*)

Registers chosen inline handler.

Параметры

- `callback` (`function`) – function to be called
- `func` (`function`) – Function executed as a filter
- `pass_bot` (`bool`) – True if you need to pass TeleBot instance to handler(useful for separating handlers into different files)
- `kwargs` – Optional keyword arguments(custom filters)

Результат
None

`register_edited_channel_post_handler`(*callback: Callable*, *content_ types: Optional[List[str]] = None*, *commands: Optional[List[str]] = None*, *regexp: Optional[str] = None*, *func: Optional[Callable] = None*, *pass_ bot: Optional[bool] = False*, ***kwargs*)

Registers edited channel post message handler.

Параметры

- `callback` (`function`) – function to be called
- `content_types` (`list` of `str`) – Supported message content types. Must be a list. Defaults to [„text“].
- `commands` (`list` of `str`) – list of commands
- `regexp` (`str`) – Regular expression
- `func` (`function`) – Function executed as a filter
- `pass_bot` (`bool`) – True if you need to pass TeleBot instance to handler(useful for separating handlers into different files)
- `kwargs` – Optional keyword arguments(custom filters)

Результат
decorated function

`register_edited_message_handler(`*callback: Callable*, *content_types: Optional[List[str]] = None*, *commands: Optional[List[str]] = None*, *regexp: Optional[str] = None*, *func: Optional[Callable] = None*, *chat_types: Optional[List[str]] = None*, *pass_bot: Optional[bool] = False*, ***kwargs*`)`

Registers edited message handler.

Параметры

- `callback` (`function`) – function to be called
- `content_types` (`list` of `str`) – Supported message content types. Must be a list. Defaults to [„text“].
- `commands` (`list` of `str`) – list of commands
- `regexp` (`str`) – Regular expression
- `func` (`function`) – Function executed as a filter
- `chat_types` (`bool`) – True for private chat
- `pass_bot` (`bool`) – True if you need to pass TeleBot instance to handler(useful for separating handlers into different files)
- `kwargs` – Optional keyword arguments(custom filters)

Результат
None

`register_for_reply(`*message:* Message, *callback: Callable*, **args*, ***kwargs*`)` → None

Registers a callback function to be notified when a reply to *message* arrives.

Warning: In case *callback* as lambda function, saving reply handlers will not work.

Параметры

- `message` (*`telebot.types.Message`*) – The message for which we are awaiting a reply.
- `callback` (`Callable[[telebot.types.Message], None]`) – The callback function to be called when a reply arrives. Must accept one *message* parameter, which will contain the replied message.
- `args` – Optional arguments for the callback function.
- `kwargs` – Optional keyword arguments for the callback function.

Результат
None

`register_for_reply_by_message_id(`*message_id: int*, *callback: Callable*, **args*, ***kwargs*`)` → None

Registers a callback function to be notified when a reply to *message* arrives.

Warning: In case *callback* as lambda function, saving reply handlers will not work.

Параметры

- `message_id` (`int`) – The id of the message for which we are awaiting a reply.
- `callback` (`Callable[[telebot.types.Message], None]`) – The callback function to be called when a reply arrives. Must accept one *message* parameter, which will contain the replied message.
- `args` – Optional arguments for the callback function.

- `kwargs` – Optional keyword arguments for the callback function.

Результат
None

`register_inline_handler(`*callback: Callable*, *func: Callable*, *pass_ bot: Optional[bool] = False*, ***kwargs*`)`

Registers inline handler.

Параметры

- `callback` (`function`) – function to be called
- `func` (`function`) – Function executed as a filter
- `pass_bot` (`bool`) – True if you need to pass TeleBot instance to handler(useful for separating handlers into different files)
- `kwargs` – Optional keyword arguments(custom filters)

Результат
decorated function

`register_message_handler(`*callback: Callable*, *content_ types: Optional[List[str]] = None*, *commands: Optional[List[str]] = None*, *regexp: Optional[str] = None*, *func: Optional[Callable] = None*, *chat_ types: Optional[List[str]] = None*, *pass_ bot: Optional[bool] = False*, ***kwargs*`)`

Registers message handler.

Параметры

- `callback` (`function`) – function to be called
- `content_types` (`list` of `str`) – Supported message content types. Must be a list. Defaults to [„text“].
- `commands` (`list` of `str`) – list of commands
- `regexp` (`str`) –
- `func` (`function`) – Function executed as a filter
- `chat_types` (`list` of `str`) – List of chat types
- `pass_bot` (`bool`) – True if you need to pass TeleBot instance to handler(useful for separating handlers into different files)
- `kwargs` – Optional keyword arguments(custom filters)

Результат
None

`register_middleware_handler(`*callback*, *update_ types=None*`)`

Adds function-based middleware handler.

This function will register your decorator function. Function-based middlewares are executed before handlers. But, be careful and check type of the update inside the handler if more than one update_type is given

Example:

bot = TeleBot(„TOKEN“)

bot.register_middleware_handler(print_channel_post_text, update_types=[„channel_post“, „edited_channel_post“])

Параметры

- `callback` (`function`) – Function that will be used as a middleware handler.
- `update_types` (`list` of `str`) – Optional list of update types that can be passed into the middleware handler.

Результат
None

`register_my_chat_member_handler`(*callback: Callable*, *func: Optional[Callable] = None*, *pass_bot: Optional[bool] = False*, ***kwargs*)

Registers my chat member handler.

Параметры

- `callback` (`function`) – function to be called
- `func` (`function`) – Function executed as a filter
- `pass_bot` (`bool`) – True if you need to pass TeleBot instance to handler(useful for separating handlers into different files)
- `kwargs` – Optional keyword arguments(custom filters)

Результат
None

`register_next_step_handler`(*message:* Message, *callback: Callable*, **args*, ***kwargs*) → None

Registers a callback function to be notified when new message arrives after *message*.

Warning: In case *callback* as lambda function, saving next step handlers will not work.

Параметры

- `message` (*`telebot.types.Message`*) – The message for which we want to handle new message in the same chat.
- `callback` (`Callable[[telebot.types.Message], None]`) – The callback function which next new message arrives.
- `args` – Args to pass in callback func
- `kwargs` – Args to pass in callback func

Результат
None

`register_next_step_handler_by_chat_id`(*chat_id: Union[int, str]*, *callback: Callable*, **args*, ***kwargs*) → None

Registers a callback function to be notified when new message arrives after *message*.

Warning: In case *callback* as lambda function, saving next step handlers will not work.

Параметры

- `chat_id` (`int` or `str`) – The chat for which we want to handle new message.
- `callback` (`Callable[[telebot.types.Message], None]`) – The callback function which next new message arrives.
- `args` – Args to pass in callback func
- `kwargs` – Args to pass in callback func

Результат
None

`register_poll_answer_handler(`*callback: Callable*, *func: Callable*, *pass_ bot: Optional[bool] = False*, ***kwargs*`)`

Registers poll answer handler.

Параметры

- `callback` (`function`) – function to be called
- `func` (`function`) – Function executed as a filter
- `pass_bot` (`bool`) – True if you need to pass TeleBot instance to handler(useful for separating handlers into different files)
- `kwargs` – Optional keyword arguments(custom filters)

Результат
None

`register_poll_handler(`*callback: Callable*, *func: Callable*, *pass_ bot: Optional[bool] = False*, ***kwargs*`)`

Registers poll handler.

Параметры

- `callback` (`function`) – function to be called
- `func` (`function`) – Function executed as a filter
- `pass_bot` (`bool`) – True if you need to pass TeleBot instance to handler(useful for separating handlers into different files)
- `kwargs` – Optional keyword arguments(custom filters)

Результат
None

`register_pre_checkout_query_handler(`*callback: Callable*, *func: Callable*, *pass_ bot: Optional[bool] = False*, ***kwargs*`)`

Registers pre-checkout request handler.

Параметры

- `callback` (`function`) – function to be called
- `func` – Function executed as a filter
- `pass_bot` (`bool`) – True if you need to pass TeleBot instance to handler(useful for separating handlers into different files)
- `kwargs` – Optional keyword arguments(custom filters)

Результат
decorated function

`register_shipping_query_handler(`*callback: Callable*, *func: Callable*, *pass_ bot: Optional[bool] = False*, ***kwargs*`)`

Registers shipping query handler.

Параметры

- `callback` (`function`) – function to be called

- `func` (`function`) – Function executed as a filter
- `pass_bot` (`bool`) – True if you need to pass TeleBot instance to handler(useful for separating handlers into different files)
- `kwargs` – Optional keyword arguments(custom filters)

Результат
None

`remove_webhook()` → bool

Deletes webhooks using set_webhook() function.

Результат
True on success.

Тип результата
`bool`

`reopen_forum_topic(`*chat_id: Union[str, int]*, *message_thread_id: int*`)` → bool

Use this method to reopen a closed topic in a forum supergroup chat. The bot must be an administrator in the chat for this to work and must have the can_manage_topics administrator rights, unless it is the creator of the topic. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#reopenforumtopic

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `message_thread_id` (`int`) – Identifier of the topic to reopen

Результат
On success, True is returned.

Тип результата
`bool`

`reply_to(`*message:* Message, *text: str*, ***kwargs*`)` → *Message*

Convenience function for *send_message(message.chat.id, text, reply_to_message_id=message.message_id, **kwargs)*

Параметры

- `message` (*types.Message*) – Instance of *telebot.types.Message*
- `text` (`str`) – Text of the message.
- `kwargs` – Additional keyword arguments which are passed to *telebot.TeleBot.send_message()*

Результат
On success, the sent Message is returned.

Тип результата
telebot.types.Message

`reset_data(`*user_id: int*, *chat_id: Optional[int] = None*`)`

Reset data for a user in chat.

Параметры

- `user_id` (`int`) – User's identifier

- `chat_id` (`int`) – Chat's identifier

Результат
None

`restrict_chat_member`(*chat_id: Union[int, str]*, *user_id: int*, *until_date: Optional[Union[int, datetime]] = None*, *can_send_messages: Optional[bool] = None*, *can_send_media_messages: Optional[bool] = None*, *can_send_polls: Optional[bool] = None*, *can_send_other_messages: Optional[bool] = None*, *can_add_web_page_previews: Optional[bool] = None*, *can_change_info: Optional[bool] = None*, *can_invite_users: Optional[bool] = None*, *can_pin_messages: Optional[bool] = None*) → bool

Use this method to restrict a user in a supergroup. The bot must be an administrator in the supergroup for this to work and must have the appropriate admin rights. Pass True for all boolean parameters to lift restrictions from a user.

Telegram documentation: https://core.telegram.org/bots/api#restrictchatmember

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target group or username of the target supergroup or channel (in the format @channelusername)
- `user_id` (`int`) – Unique identifier of the target user
- `until_date` (`int` or `datetime`) – Date when restrictions will be lifted for the user, unix time. If user is restricted for more than 366 days or less than 30 seconds from the current time, they are considered to be restricted forever
- `can_send_messages` (`bool`) – Pass True, if the user can send text messages, contacts, locations and venues
- `can_send_media_messages` (`bool`) – Pass True, if the user can send audios, documents, photos, videos, video notes and voice notes, implies can_send_messages
- `can_send_polls` (`bool`) – Pass True, if the user is allowed to send polls, implies can_send_messages
- `can_send_other_messages` (`bool`) – Pass True, if the user can send animations, games, stickers and use inline bots, implies can_send_media_messages
- `can_add_web_page_previews` (`bool`) – Pass True, if the user may add web page previews to their messages, implies can_send_media_messages
- `can_change_info` (`bool`) – Pass True, if the user is allowed to change the chat title, photo and other settings. Ignored in public supergroups
- `can_invite_users` (`bool`) – Pass True, if the user is allowed to invite new users to the chat, implies can_invite_users
- `can_pin_messages` (`bool`) – Pass True, if the user is allowed to pin messages. Ignored in public supergroups

Результат
True on success

Тип результата
`bool`

`retrieve_data`(*user_id: int*, *chat_id: Optional[int] = None*) → Optional[Any]

Returns context manager with data for a user in chat.

Параметры

- `user_id` (*int*) – User identifier
- `chat_id` (*int, optional*) – Chat's unique identifier, defaults to user_id

Результат
Context manager with data for a user in chat

Тип результата
Optional[Any]

`revoke_chat_invite_link`(*chat_id: Union[int, str]*, *invite_link: str*) → *ChatInviteLink*

Use this method to revoke an invite link created by the bot. Note: If the primary link is revoked, a new link is automatically generated The bot must be an administrator in the chat for this to work and must have the appropriate admin rights.

Telegram documentation: https://core.telegram.org/bots/api#revokechatinvitelink

Параметры

- `chat_id` (`int` or `str`) – Id: Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `invite_link` (`str`) – The invite link to revoke

Результат
Returns the new invite link as ChatInviteLink object.

Тип результата
telebot.types.ChatInviteLink

`run_webhooks`(*listen: Optional[str] = '127.0.0.1'*, *port: Optional[int] = 443*, *url_path: Optional[str] = None*, *certificate: Optional[str] = None*, *certificate_key: Optional[str] = None*, *webhook_url: Optional[str] = None*, *max_connections: Optional[int] = None*, *allowed_updates: Optional[List] = None*, *ip_address: Optional[str] = None*, *drop_pending_updates: Optional[bool] = None*, *timeout: Optional[int] = None*, *secret_token: Optional[str] = None*, *secret_token_length: Optional[int] = 20*)

This class sets webhooks and listens to a given url and port.

Requires fastapi, uvicorn, and latest version of starlette.

Параметры

- `listen` (`str`, optional) – IP address to listen to, defaults to «127.0.0.1»
- `port` (`int`, optional) – A port which will be used to listen to webhooks., defaults to 443
- `url_path` (`str`, optional) – Path to the webhook. Defaults to /token, defaults to None
- `certificate` (`str`, optional) – Path to the certificate file, defaults to None
- `certificate_key` (`str`, optional) – Path to the certificate key file, defaults to None
- `webhook_url` (`str`, optional) – Webhook URL to be set, defaults to None
- `max_connections` (`int`, optional) – Maximum allowed number of simultaneous HTTPS connections to the webhook for update delivery, 1-100. Defaults to 40. Use lower values to limit the load on your bot's server, and higher values to increase your bot's throughput., defaults to None

- `allowed_updates` (`list`, optional) – A JSON-serialized list of the update types you want your bot to receive. For example, specify [“message”, “edited_channel_post”, “callback_query”] to only receive updates of these types. See Update for a complete list of available update types. Specify an empty list to receive all updates regardless of type (default). If not specified, the previous setting will be used. defaults to None
- `ip_address` (`str`, optional) – The fixed IP address which will be used to send webhook requests instead of the IP address resolved through DNS, defaults to None
- `drop_pending_updates` (`bool`, optional) – Pass True to drop all pending updates, defaults to None
- `timeout` (`int`, optional) – Request connection timeout, defaults to None
- `secret_token` (`str`, optional) – Secret token to be used to verify the webhook request, defaults to None
- `secret_token_length` (`int`, optional) – Length of a secret token, defaults to 20

Исключение

`ImportError` – If necessary libraries were not installed.

`send_animation`(*chat_id: Union[int, str]*, *animation: Union[Any, str]*, *duration: Optional[int] = None*, *width: Optional[int] = None*, *height: Optional[int] = None*, *thumb: Optional[Union[str, Any]] = None*, *caption: Optional[str] = None*, *parse_mode: Optional[str] = None*, *caption_entities: Optional[List[MessageEntity]] = None*, *disable_notification: Optional[bool] = None*, *protect_content: Optional[bool] = None*, *reply_to_message_id: Optional[int] = None*, *allow_sending_without_reply: Optional[bool] = None*, *reply_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None*, *timeout: Optional[int] = None*, *message_thread_id: Optional[int] = None*) → *Message*

Use this method to send animation files (GIF or H.264/MPEG-4 AVC video without sound). On success, the sent Message is returned. Bots can currently send animation files of up to 50 MB in size, this limit may be changed in the future.

Telegram documentation: https://core.telegram.org/bots/api#sendanimation

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `animation` (`str` or *`telebot.types.InputFile`*) – Animation to send. Pass a file_id as String to send an animation that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get an animation from the Internet, or upload a new animation using multipart/form-data.
- `duration` (`int`) – Duration of sent animation in seconds
- `width` (`int`) – Animation width
- `height` (`int`) – Animation height
- `thumb` (`str` or *`telebot.types.InputFile`*) – Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail’s width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can’t be reused and can be only uploaded as a new file, so you can pass “attach://<file_attach_name>” if the thumbnail was uploaded using multipart/form-data under <file_attach_name>.

- `caption` (`str`) – Animation caption (may also be used when resending animation by file_id), 0-1024 characters after entities parsing
- `parse_mode` (`str`) – Mode for parsing entities in the animation caption
- `protect_content` (`bool`) – Protects the contents of the sent message from forwarding and saving
- `reply_to_message_id` (`int`) – If the message is a reply, ID of the original message
- `reply_markup` (*telebot.types.InlineKeyboardMarkup* or *telebot.types.ReplyKeyboardMarkup* or *telebot.types.ReplyKeyboardRemove* or *telebot.types.ForceReply*) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user.
- `disable_notification` (`bool`) – Sends the message silently. Users will receive a notification with no sound.
- `timeout` (`int`) – Timeout in seconds for the request.
- `caption_entities` (`list` of *telebot.types.MessageEntity*) – List of special entities that appear in the caption, which can be specified instead of parse_mode
- `allow_sending_without_reply` (`bool`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- `message_thread_id` (`int`) – Identifier of a message thread, in which the video will be sent

Результат
On success, the sent Message is returned.

Тип результата
telebot.types.Message

`send_audio`(*chat_id: Union[int, str]*, *audio: Union[Any, str]*, *caption: Optional[str] = None*, *duration: Optional[int] = None*, *performer: Optional[str] = None*, *title: Optional[str] = None*, *reply_to_message_id: Optional[int] = None*, *reply_markup: Optional[Union[*InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply*]] = None*, *parse_mode: Optional[str] = None*, *disable_notification: Optional[bool] = None*, *timeout: Optional[int] = None*, *thumb: Optional[Union[str, Any]] = None*, *caption_entities: Optional[List[*MessageEntity*]] = None*, *allow_sending_without_reply: Optional[bool] = None*, *protect_content: Optional[bool] = None*, *message_thread_id: Optional[int] = None*) → *Message*

Use this method to send audio files, if you want Telegram clients to display them in the music player. Your audio must be in the .MP3 or .M4A format. On success, the sent Message is returned. Bots can currently send audio files of up to 50 MB in size, this limit may be changed in the future.

For sending voice messages, use the send_voice method instead.

Telegram documentation: https://core.telegram.org/bots/api#sendaudio

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `audio` (`str` or *telebot.types.InputFile*) – Audio file to send. Pass a file_id as String to send an audio file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get an audio file from the Internet,

or upload a new one using multipart/form-data. Audio must be in the .MP3 or .M4A format.

- `caption` (`str`) – Audio caption, 0-1024 characters after entities parsing
- `duration` (`int`) – Duration of the audio in seconds
- `performer` (`str`) – Performer
- `title` (`str`) – Track name
- `reply_to_message_id` (`int`) – If the message is a reply, ID of the original message
- `reply_markup` (*telebot.types.InlineKeyboardMarkup* or *telebot.types.ReplyKeyboardMarkup* or *telebot.types.ReplyKeyboardRemove* or *telebot.types.ForceReply*) –
- `parse_mode` (`str`) – Mode for parsing entities in the audio caption. See formatting options for more details.
- `disable_notification` (`bool`) – Sends the message silently. Users will receive a notification with no sound.
- `timeout` (`int`) – Timeout in seconds for the request.
- `thumb` (`str`) – Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass "attach://<file_attach_name>" if the thumbnail was uploaded using multipart/form-data under <file_attach_name>
- `caption_entities` (`list` of *telebot.types.MessageEntity*) – A JSON-serialized list of special entities that appear in the caption, which can be specified instead of parse_mode
- `allow_sending_without_reply` (`bool`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- `protect_content` (`bool`) – Protects the contents of the sent message from forwarding and saving
- `message_thread_id` (`int`) – Identifier of a message thread, in which the message will be sent

Результат
On success, the sent Message is returned.

Тип результата
telebot.types.Message

`send_chat_action`(*chat_id: Union[int, str]*, *action: str*, *timeout: Optional[int] = None*) → bool

Use this method when you need to tell the user that something is happening on the bot's side. The status is set for 5 seconds or less (when a message arrives from your bot, Telegram clients clear its typing status). Returns True on success.

Example: The ImageBot needs some time to process a request and upload the image. Instead of sending a text message along the lines of "Retrieving image, please wait...", the bot may use sendChatAction with action = upload_photo. The user will see a "sending photo" status for the bot.

Telegram documentation: https://core.telegram.org/bots/api#sendchataction

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel
- `action` (`str`) – Type of action to broadcast. Choose one, depending on what the user is about to receive: typing for text messages, upload_photo for photos, record_video or upload_video for videos, record_voice or upload_voice for voice notes, upload_document for general files, choose_sticker for stickers, find_location for location data, record_video_note or upload_video_note for video notes.
- `timeout` (`int`) – Timeout in seconds for the request.

Результат
Returns True on success.

Тип результата
`bool`

`send_contact`(*chat_id: Union[int, str]*, *phone_number: str*, *first_name: str*, *last_name: Optional[str] = None*, *vcard: Optional[str] = None*, *disable_notification: Optional[bool] = None*, *reply_to_message_id: Optional[int] = None*, *reply_markup: Optional[Union[*InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply*]] = None*, *timeout: Optional[int] = None*, *allow_sending_without_reply: Optional[bool] = None*, *protect_content: Optional[bool] = None*, *message_thread_id: Optional[int] = None*) → *Message*

Use this method to send phone contacts. On success, the sent Message is returned.

Telegram documentation: https://core.telegram.org/bots/api#sendcontact

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel
- `phone_number` (`str`) – Contact's phone number
- `first_name` (`str`) – Contact's first name
- `last_name` (`str`) – Contact's last name
- `vcard` (`str`) – Additional data about the contact in the form of a vCard, 0-2048 bytes
- `disable_notification` (`bool`) – Sends the message silently. Users will receive a notification with no sound.
- `reply_to_message_id` (`int`) – If the message is a reply, ID of the original message
- `reply_markup` (*telebot.types.InlineKeyboardMarkup* or *telebot.types.ReplyKeyboardMarkup* or *telebot.types.ReplyKeyboardRemove* or *telebot.types.ForceReply*) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user.
- `timeout` (`int`) – Timeout in seconds for the request.
- `allow_sending_without_reply` (`bool`) – Pass True, if the message should be sent even if one of the specified replied-to messages is not found.
- `protect_content` (`bool`) – Protects the contents of the sent message from forwarding and saving

- `message_thread_id` (`int`) – The thread identifier of a message from which the reply will be sent

Результат
On success, the sent Message is returned.

Тип результата
telebot.types.Message

`send_dice`(*chat_id: Union[int, str]*, *emoji: Optional[str] = None*, *disable_notification: Optional[bool] = None*, *reply_to_message_id: Optional[int] = None*, *reply_markup: Optional[Union[*InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply*]] = None*, *timeout: Optional[int] = None*, *allow_sending_without_reply: Optional[bool] = None*, *protect_content: Optional[bool] = None*, *message_thread_id: Optional[int] = None*) → *Message*

Use this method to send an animated emoji that will display a random value. On success, the sent Message is returned.

Telegram documentation: https://core.telegram.org/bots/api#senddice

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `emoji` (`str`) – Emoji on which the dice throw animation is based. Currently, must be one of “”, “”, “”, “”, “”, or “”. Dice can have values 1-6 for “”, “” and “”, values 1-5 for “” and “”, and values 1-64 for “”. Defaults to “”
- `disable_notification` (`bool`) – Sends the message silently. Users will receive a notification with no sound.
- `reply_to_message_id` (`int`) – If the message is a reply, ID of the original message
- `reply_markup` (*telebot.types.InlineKeyboardMarkup* or *telebot.types.ReplyKeyboardMarkup* or *telebot.types.ReplyKeyboardRemove* or *telebot.types.ForceReply*) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user.
- `timeout` (`int`) – Timeout in seconds for the request.
- `allow_sending_without_reply` (`bool`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- `protect_content` (`bool`) – Protects the contents of the sent message from forwarding
- `message_thread_id` (`int`) – Identifier of a message thread, in which the message will be sent

Результат
On success, the sent Message is returned.

Тип результата
telebot.types.Message

send_document(*chat_id: Union[int, str]*, *document: Union[Any, str]*, *reply_to_message_id: Optional[int] = None*, *caption: Optional[str] = None*, *reply_markup: Optional[Union[*InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply*]] = None*, *parse_mode: Optional[str] = None*, *disable_notification: Optional[bool] = None*, *timeout: Optional[int] = None*, *thumb: Optional[Union[str, Any]] = None*, *caption_entities: Optional[List[*MessageEntity*]] = None*, *allow_sending_without_reply: Optional[bool] = None*, *visible_file_name: Optional[str] = None*, *disable_content_type_detection: Optional[bool] = None*, *data: Optional[Union[str, Any]] = None*, *protect_content: Optional[bool] = None*, *message_thread_id: Optional[int] = None*) → *Message*

Use this method to send general files.

Telegram documentation: https://core.telegram.org/bots/api#senddocument

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `document` (`str` or *`telebot.types.InputFile`*) – (document) File to send. Pass a file_id as String to send a file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a file from the Internet, or upload a new one using multipart/form-data
- `reply_to_message_id` (`int`) – If the message is a reply, ID of the original message
- `caption` (`str`) – Document caption (may also be used when resending documents by file_id), 0-1024 characters after entities parsing
- `reply_markup` (*`telebot.types.InlineKeyboardMarkup`* or *`telebot.types.ReplyKeyboardMarkup`* or *`telebot.types.ReplyKeyboardRemove`* or *`telebot.types.ForceReply`*) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user.
- `parse_mode` (`str`) – Mode for parsing entities in the document caption
- `disable_notification` (`bool`) – Sends the message silently. Users will receive a notification with no sound.
- `timeout` (`int`) – Timeout in seconds for the request.
- `thumb` (`str` or *`telebot.types.InputFile`*) – InputFile or String : Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass "attach://<file_attach_name>" if the thumbnail was uploaded using multipart/form-data under <file_attach_name>
- `caption_entities` (`list` of *`telebot.types.MessageEntity`*) – A JSON-serialized list of special entities that appear in the caption, which can be specified instead of parse_mode
- `allow_sending_without_reply` (`bool`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- `visible_file_name` (`str`) – allows to define file name that will be visible in the Telegram instead of original file name

- `disable_content_type_detection` (`bool`) – Disables automatic server-side content type detection for files uploaded using multipart/form-data
- `data` (`str`) – function typo miss compatibility: do not use it
- `protect_content` (`bool`) – Protects the contents of the sent message from forwarding and saving
- `message_thread_id` (`int`) – The thread to which the message will be sent

Результат
On success, the sent Message is returned.

Тип результата
telebot.types.Message

`send_game`(*chat_id: Union[int, str]*, *game_short_name: str*, *disable_notification: Optional[bool] = None*, *reply_to_message_id: Optional[int] = None*, *reply_markup: Optional[Union[*InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply*]] = None*, *timeout: Optional[int] = None*, *allow_sending_without_reply: Optional[bool] = None*, *protect_content: Optional[bool] = None*, *message_thread_id: Optional[int] = None*) → *Message*

Used to send the game.

Telegram documentation: https://core.telegram.org/bots/api#sendgame

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `game_short_name` (`str`) – Short name of the game, serves as the unique identifier for the game. Set up your games via @BotFather.
- `disable_notification` (`bool`) – Sends the message silently. Users will receive a notification with no sound.
- `reply_to_message_id` (`int`) – If the message is a reply, ID of the original message
- `reply_markup` (`InlineKeyboardMarkup` or `ReplyKeyboardMarkup` or `ReplyKeyboardRemove` or `ForceReply`) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user.
- `timeout` (`int`) – Timeout in seconds for waiting for a response from the bot.
- `allow_sending_without_reply` (`bool`) – Pass True, if the message should be sent even if one of the specified replied-to messages is not found.
- `protect_content` (`bool`) – Pass True, if content of the message needs to be protected from being viewed by the bot.
- `message_thread_id` (`int`) – The identifier of a message thread, in which the game message will be sent.

Результат
On success, the sent Message is returned.

Тип результата
types.Message

send_invoice(*chat_id: Union[int, str]*, *title: str*, *description: str*, *invoice_payload: str*, *provider_token: str*, *currency: str*, *prices: List[LabeledPrice]*, *start_parameter: Optional[str] = None*, *photo_url: Optional[str] = None*, *photo_size: Optional[int] = None*, *photo_width: Optional[int] = None*, *photo_height: Optional[int] = None*, *need_name: Optional[bool] = None*, *need_phone_number: Optional[bool] = None*, *need_email: Optional[bool] = None*, *need_shipping_address: Optional[bool] = None*, *send_phone_number_to_provider: Optional[bool] = None*, *send_email_to_provider: Optional[bool] = None*, *is_flexible: Optional[bool] = None*, *disable_notification: Optional[bool] = None*, *reply_to_message_id: Optional[int] = None*, *reply_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None*, *provider_data: Optional[str] = None*, *timeout: Optional[int] = None*, *allow_sending_without_reply: Optional[bool] = None*, *max_tip_amount: Optional[int] = None*, *suggested_tip_amounts: Optional[List[int]] = None*, *protect_content: Optional[bool] = None*, *message_thread_id: Optional[int] = None*) → *Message*

Sends invoice.

Telegram documentation: https://core.telegram.org/bots/api#sendinvoice

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target private chat
- `title` (`str`) – Product name, 1-32 characters
- `description` (`str`) – Product description, 1-255 characters
- `invoice_payload` (`str`) – Bot-defined invoice payload, 1-128 bytes. This will not be displayed to the user, use for your internal processes.
- `provider_token` (`str`) – Payments provider token, obtained via @Botfather
- `currency` (`str`) – Three-letter ISO 4217 currency code, see https://core.telegram.org/bots/payments#supported-currencies
- `prices` (List[*types.LabeledPrice*]) – Price breakdown, a list of components (e.g. product price, tax, discount, delivery cost, delivery tax, bonus, etc.)
- `start_parameter` (`str`) – Unique deep-linking parameter that can be used to generate this invoice when used as a start parameter
- `photo_url` (`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.
- `photo_size` (`int`) – Photo size in bytes
- `photo_width` (`int`) – Photo width
- `photo_height` (`int`) – Photo height
- `need_name` (`bool`) – Pass True, if you require the user's full name to complete the order
- `need_phone_number` (`bool`) – Pass True, if you require the user's phone number to complete the order
- `need_email` (`bool`) – Pass True, if you require the user's email to complete the order
- `need_shipping_address` (`bool`) – Pass True, if you require the user's shipping address to complete the order

- `is_flexible` (`bool`) – Pass True, if the final price depends on the shipping method
- `send_phone_number_to_provider` (`bool`) – Pass True, if user's phone number should be sent to provider
- `send_email_to_provider` (`bool`) – Pass True, if user's email address should be sent to provider
- `disable_notification` (`bool`) – Sends the message silently. Users will receive a notification with no sound.
- `reply_to_message_id` (`int`) – If the message is a reply, ID of the original message
- `reply_markup` (`str`) – A JSON-serialized object for an inline keyboard. If empty, one „Pay total price" button will be shown. If not empty, the first button must be a Pay button
- `provider_data` (`str`) – A JSON-serialized data about the invoice, which will be shared with the payment provider. A detailed description of required fields should be provided by the payment provider.
- `timeout` (`int`) – Timeout of a request, defaults to None
- `allow_sending_without_reply` (`bool`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- `max_tip_amount` (`int`) – The maximum accepted amount for tips in the smallest units of the currency
- `suggested_tip_amounts` (`list` of `int`) – A JSON-serialized array of suggested amounts of tips in the smallest units of the currency. At most 4 suggested tip amounts can be specified. The suggested tip amounts must be positive, passed in a strictly increased order and must not exceed max_tip_amount.
- `protect_content` (`bool`) – Protects the contents of the sent message from forwarding and saving
- `message_thread_id` (`int`) – The identifier of a message thread, in which the invoice message will be sent

Результат
On success, the sent Message is returned.

Тип результата
types.Message

`send_location`(*chat_id: Union[int, str]*, *latitude: float*, *longitude: float*, *live_period: Optional[int] = None*, *reply_to_message_id: Optional[int] = None*, *reply_markup: Optional[Union[*InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply*]] = None*, *disable_notification: Optional[bool] = None*, *timeout: Optional[int] = None*, *horizontal_accuracy: Optional[float] = None*, *heading: Optional[int] = None*, *proximity_alert_radius: Optional[int] = None*, *allow_sending_without_reply: Optional[bool] = None*, *protect_content: Optional[bool] = None*, *message_thread_id: Optional[int] = None*) → *Message*

Use this method to send point on the map. On success, the sent Message is returned.

Telegram documentation: https://core.telegram.org/bots/api#sendlocation

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)

- `latitude` (`float`) – Latitude of the location
- `longitude` (`float`) – Longitude of the location
- `live_period` (`int`) – Period in seconds for which the location will be updated (see Live Locations, should be between 60 and 86400.
- `reply_to_message_id` (`int`) – If the message is a reply, ID of the original message
- `reply_markup` (*telebot.types.InlineKeyboardMarkup* or *telebot.types.ReplyKeyboardMarkup* or *telebot.types.ReplyKeyboardRemove* or *telebot.types.ForceReply*) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user.
- `disable_notification` (`bool`) – Sends the message silently. Users will receive a notification with no sound.
- `timeout` (`int`) – Timeout in seconds for the request.
- `horizontal_accuracy` (`float`) – The radius of uncertainty for the location, measured in meters; 0-1500
- `heading` (`int`) – For live locations, a direction in which the user is moving, in degrees. Must be between 1 and 360 if specified.
- `proximity_alert_radius` (`int`) – For live locations, a maximum distance for proximity alerts about approaching another chat member, in meters. Must be between 1 and 100000 if specified.
- `allow_sending_without_reply` (`bool`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- `protect_content` (`bool`) – Protects the contents of the sent message from forwarding and saving
- `message_thread_id` (`int`) – Identifier of a message thread, in which the message will be sent

Результат

On success, the sent Message is returned.

Тип результата

telebot.types.Message

`send_media_group`(*chat_id: Union[int, str]*, *media: List[Union[*InputMediaAudio, InputMediaDocument, InputMediaPhoto, InputMediaVideo*]]*, *disable_notification: Optional[bool] = None*, *protect_content: Optional[bool] = None*, *reply_to_message_id: Optional[int] = None*, *timeout: Optional[int] = None*, *allow_sending_without_reply: Optional[bool] = None*, *message_thread_id: Optional[int] = None*) → List[*Message*]

Use this method to send a group of photos, videos, documents or audios as an album. Documents and audio files can be only grouped in an album with messages of the same type. On success, an array of Messages that were sent is returned.

Telegram documentation: https://core.telegram.org/bots/api#sendmediagroup

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)

- `media` (`list` of *types.InputMedia*) – A JSON-serialized array describing messages to be sent, must include 2-10 items
- `disable_notification` (`bool`) – Sends the messages silently. Users will receive a notification with no sound.
- `protect_content` (`bool`) – Protects the contents of the sent message from forwarding and saving
- `reply_to_message_id` (`int`) – If the message is a reply, ID of the original message
- `timeout` (`int`) – Timeout in seconds for the request.
- `allow_sending_without_reply` (`bool`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- `message_thread_id` (`int`) – Identifier of a message thread, in which the media group will be sent

Результат
On success, an array of Messages that were sent is returned.

Тип результата
List[*types.Message*]

`send_message`(*chat_id: Union[int, str]*, *text: str*, *parse_mode: Optional[str] = None*, *entities: Optional[List[MessageEntity]] = None*, *disable_web_page_preview: Optional[bool] = None*, *disable_notification: Optional[bool] = None*, *protect_content: Optional[bool] = None*, *reply_to_message_id: Optional[int] = None*, *allow_sending_without_reply: Optional[bool] = None*, *reply_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None*, *timeout: Optional[int] = None*, *message_thread_id: Optional[int] = None*) → *Message*

Use this method to send text messages.

Warning: Do not send more than about 4096 characters each message, otherwise you'll risk an HTTP 414 error. If you must send more than 4096 characters, use the *split_string* or *smart_split* function in util.py.

Telegram documentation: https://core.telegram.org/bots/api#sendmessage

Параметры

- `chat_id` (`int` or `str`) – 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
- `parse_mode` (`str`) – Mode for parsing entities in the message text.
- `entities` (Array of *telebot.types.MessageEntity*) – List of special entities that appear in message text, which can be specified instead of parse_mode
- `disable_web_page_preview` (`bool`) – Disables link previews for links in this message
- `disable_notification` (`bool`) – Sends the message silently. Users will receive a notification with no sound.
- `protect_content` (`bool`) – If True, the message content will be hidden for all users except for the target user
- `reply_to_message_id` (`int`) – If the message is a reply, ID of the original message

- `allow_sending_without_reply` (`bool`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- `reply_markup` (*telebot.types.InlineKeyboardMarkup* or *telebot.types.ReplyKeyboardMarkup* or *telebot.types.ReplyKeyboardRemove* or *telebot.types.ForceReply*) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user.
- `timeout` (`int`) – Timeout in seconds for the request.
- `message_thread_id` (`int`) – Identifier of a message thread, in which the message will be sent

Результат
On success, the sent Message is returned.

Тип результата
telebot.types.Message

`send_photo`(*chat_id: Union[int, str]*, *photo: Union[Any, str]*, *caption: Optional[str] = None*, *parse_mode: Optional[str] = None*, *caption_entities: Optional[List[MessageEntity]] = None*, *disable_notification: Optional[bool] = None*, *protect_content: Optional[bool] = None*, *reply_to_message_id: Optional[int] = None*, *allow_sending_without_reply: Optional[bool] = None*, *reply_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None*, *timeout: Optional[int] = None*, *message_thread_id: Optional[int] = None*) → *Message*

Use this method to send photos. On success, the sent Message is returned.

Telegram documentation: https://core.telegram.org/bots/api#sendphoto

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `photo` (`str` or *telebot.types.InputFile*) – Photo to send. Pass a file_id as String to send a photo that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a photo from the Internet, or upload a new photo using multipart/form-data. The photo must be at most 10 MB in size. The photo's width and height must not exceed 10000 in total. Width and height ratio must be at most 20.
- `caption` (`str`) – Photo caption (may also be used when resending photos by file_id), 0-1024 characters after entities parsing
- `parse_mode` (`str`) – Mode for parsing entities in the photo caption.
- `caption_entities` (`list` of *telebot.types.MessageEntity*) – A JSON-serialized list of special entities that appear in the caption, which can be specified instead of parse_mode
- `disable_notification` (`bool`) – Sends the message silently. Users will receive a notification with no sound.
- `protect_content` (`bool`) – Protects the contents of the sent message from forwarding and saving
- `reply_to_message_id` (`int`) – If the message is a reply, ID of the original message
- `allow_sending_without_reply` (`bool`) – Pass True, if the message should be sent even if the specified replied-to message is not found

- **reply_markup** (*telebot.types.InlineKeyboardMarkup* or *telebot.types.ReplyKeyboardMarkup* or *telebot.types.ReplyKeyboardRemove* or *telebot.types.ForceReply*) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user.
- **timeout** (**int**) – Timeout in seconds for the request.
- **message_thread_id** (**int**) – Identifier of a message thread, in which the message will be sent

Результат
On success, the sent Message is returned.

Тип результата
telebot.types.Message

send_poll(*chat_id: Union[int, str]*, *question: str*, *options: List[str]*, *is_anonymous: Optional[bool] = None*, *type: Optional[str] = None*, *allows_multiple_answers: Optional[bool] = None*, *correct_option_id: Optional[int] = None*, *explanation: Optional[str] = None*, *explanation_parse_mode: Optional[str] = None*, *open_period: Optional[int] = None*, *close_date: Optional[Union[int, datetime]] = None*, *is_closed: Optional[bool] = None*, *disable_notification: Optional[bool] = False*, *reply_to_message_id: Optional[int] = None*, *reply_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None*, *allow_sending_without_reply: Optional[bool] = None*, *timeout: Optional[int] = None*, *explanation_entities: Optional[List[MessageEntity]] = None*, *protect_content: Optional[bool] = None*, *message_thread_id: Optional[int] = None*) → *Message*

Use this method to send a native poll. On success, the sent Message is returned.

Telegram documentation: https://core.telegram.org/bots/api#sendpoll

Параметры

- **chat_id** (**int** | **str**) – Unique identifier for the target chat or username of the target channel
- **question** (**str**) – Poll question, 1-300 characters
- **options** (**list** of **str**) – A JSON-serialized list of answer options, 2-10 strings 1-100 characters each
- **is_anonymous** (**bool**) – True, if the poll needs to be anonymous, defaults to True
- **type** (**str**) – Poll type, “quiz” or “regular”, defaults to “regular”
- **allows_multiple_answers** (**bool**) – True, if the poll allows multiple answers, ignored for polls in quiz mode, defaults to False
- **correct_option_id** (**int**) – 0-based identifier of the correct answer option. Available only for polls in quiz mode, defaults to None
- **explanation** (**str**) – Text that is shown when a user chooses an incorrect answer or taps on the lamp icon in a quiz-style poll, 0-200 characters with at most 2 line feeds after entities parsing
- **explanation_parse_mode** (**str**) – Mode for parsing entities in the explanation. See formatting options for more details.
- **open_period** (**int**) – Amount of time in seconds the poll will be active after creation, 5-600. Can’t be used together with close_date.

- `close_date` (`int` | `datetime`) – Point in time (Unix timestamp) when the poll will be automatically closed.
- `is_closed` (`bool`) – Pass True, if the poll needs to be immediately closed. This can be useful for poll preview.
- `disable_notification` (`bool`) – Sends the message silently. Users will receive a notification with no sound.
- `reply_to_message_id` (`int`) – If the message is a reply, ID of the original message
- `allow_sending_without_reply` (`bool`) – Pass True, if the poll allows multiple options to be voted simultaneously.
- `reply_markup` (`InlineKeyboardMarkup` | `ReplyKeyboardMarkup` | `ReplyKeyboardRemove` | `ForceReply`) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user.
- `timeout` (`int`) – Timeout in seconds for waiting for a response from the user.
- `explanation_entities` (`list` of `MessageEntity`) – A JSON-serialized list of special entities that appear in the explanation, which can be specified instead of parse_mode
- `protect_content` (`bool`) – Protects the contents of the sent message from forwarding and saving
- `message_thread_id` (`int`) – The identifier of a message thread, in which the poll will be sent

Результат
On success, the sent Message is returned.

Тип результата
types.Message

`send_sticker`(*chat_id: Union[int, str]*, *sticker: Union[Any, str]*, *reply_to_message_id: Optional[int] = None*, *reply_markup: Optional[Union[*InlineKeyboardMarkup*,* ReplyKeyboardMarkup*,* ReplyKeyboardRemove*,* ForceReply*]] = None*, *disable_notification: Optional[bool] = None*, *timeout: Optional[int] = None*, *allow_sending_without_reply: Optional[bool] = None*, *protect_content: Optional[bool] = None*, *data: Optional[Union[str, Any]] = None*, *message_thread_id: Optional[int] = None*) → *Message*

Use this method to send static .WEBP, animated .TGS, or video .WEBM stickers. On success, the sent Message is returned.

Telegram documentation: https://core.telegram.org/bots/api#sendsticker

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `sticker` (`str` or *telebot.types.InputFile*) – Sticker to send. Pass a file_id as String to send a file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a .webp file from the Internet, or upload a new one using multipart/form-data.
- `reply_to_message_id` (`int`) – If the message is a reply, ID of the original message

- `reply_markup` (*telebot.types.InlineKeyboardMarkup* or *telebot.types.ReplyKeyboardMarkup* or *telebot.types.ReplyKeyboardRemove* or *telebot.types.ForceReply*) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user.
- `disable_notification` (`bool`) – to disable the notification
- `timeout` (`int`) – Timeout in seconds for the request.
- `allow_sending_without_reply` (`bool`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- `protect_content` (`bool`) – Protects the contents of the sent message from forwarding and saving
- `data` (`str`) – function typo miss compatibility: do not use it
- `message_thread_id` (`int`) – The thread to which the message will be sent

Результат
On success, the sent Message is returned.

Тип результата
telebot.types.Message

`send_venue`(*chat_id: Union[int, str]*, *latitude: Optional[float]*, *longitude: Optional[float]*, *title: str*, *address: str*, *foursquare_id: Optional[str] = None*, *foursquare_type: Optional[str] = None*, *disable_notification: Optional[bool] = None*, *reply_to_message_id: Optional[int] = None*, *reply_markup: Optional[Union[*InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply*]] = None*, *timeout: Optional[int] = None*, *allow_sending_without_reply: Optional[bool] = None*, *google_place_id: Optional[str] = None*, *google_place_type: Optional[str] = None*, *protect_content: Optional[bool] = None*, *message_thread_id: Optional[int] = None*) → *Message*

Use this method to send information about a venue. On success, the sent Message is returned.

Telegram documentation: https://core.telegram.org/bots/api#sendvenue

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel
- `latitude` (`float`) – Latitude of the venue
- `longitude` (`float`) – Longitude of the venue
- `title` (`str`) – Name of the venue
- `address` (`str`) – Address of the venue
- `foursquare_id` (`str`) – Foursquare identifier of the venue
- `foursquare_type` (`str`) – Foursquare type of the venue, if known. (For example, “arts_entertainment/default”, “arts_entertainment/aquarium” or “food/icecream”.)
- `disable_notification` (`bool`) – Sends the message silently. Users will receive a notification with no sound.
- `reply_to_message_id` (`int`) – If the message is a reply, ID of the original message

- `reply_markup` (*telebot.types.InlineKeyboardMarkup* or *telebot.types.ReplyKeyboardMarkup* or *telebot.types.ReplyKeyboardRemove* or *telebot.types.ForceReply*) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user.
- `timeout` (`int`) – Timeout in seconds for the request.
- `allow_sending_without_reply` (`bool`) – Pass True, if the message should be sent even if one of the specified replied-to messages is not found.
- `google_place_id` (`str`) – Google Places identifier of the venue
- `google_place_type` (`str`) – Google Places type of the venue.
- `protect_content` (`bool`) – Protects the contents of the sent message from forwarding and saving
- `message_thread_id` (`int`) – The thread identifier of a message from which the reply will be sent

Результат
On success, the sent Message is returned.

Тип результата
telebot.types.Message

send_video(*chat_id: Union[int, str]*, *video: Union[Any, str]*, *duration: Optional[int] = None*, *width: Optional[int] = None*, *height: Optional[int] = None*, *thumb: Optional[Union[str, Any]] = None*, *caption: Optional[str] = None*, *parse_mode: Optional[str] = None*, *caption_entities: Optional[List[MessageEntity]] = None*, *supports_streaming: Optional[bool] = None*, *disable_notification: Optional[bool] = None*, *protect_content: Optional[bool] = None*, *reply_to_message_id: Optional[int] = None*, *allow_sending_without_reply: Optional[bool] = None*, *reply_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None*, *timeout: Optional[int] = None*, *data: Optional[Union[str, Any]] = None*, *message_thread_id: Optional[int] = None*) → *Message*

Use this method to send video files, Telegram clients support mp4 videos (other formats may be sent as Document).

Telegram documentation: https://core.telegram.org/bots/api#sendvideo

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `video` (`str` or *telebot.types.InputFile*) – Video to send. You can either pass a file_id as String to resend a video that is already on the Telegram servers, or upload a new video file using multipart/form-data.
- `duration` (`int`) – Duration of sent video in seconds
- `width` (`int`) – Video width
- `height` (`int`) – Video height
- `thumb` (`str` or *telebot.types.InputFile*) – Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using

multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass "attach://<file_attach_name>" if the thumbnail was uploaded using multipart/form-data under <file_attach_name>.

- `caption` (`str`) – Video caption (may also be used when resending videos by file_id), 0-1024 characters after entities parsing
- `parse_mode` (`str`) – Mode for parsing entities in the video caption
- `caption_entities` (`list` of *telebot.types.MessageEntity*) – List of special entities that appear in the caption, which can be specified instead of parse_mode
- `supports_streaming` (`bool`) – Pass True, if the uploaded video is suitable for streaming
- `disable_notification` (`bool`) – Sends the message silently. Users will receive a notification with no sound.
- `protect_content` (`bool`) – Protects the contents of the sent message from forwarding and saving
- `reply_to_message_id` (`int`) – If the message is a reply, ID of the original message
- `allow_sending_without_reply` (`bool`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- `reply_markup` (*telebot.types.InlineKeyboardMarkup* or *telebot.types.ReplyKeyboardMarkup* or *telebot.types.ReplyKeyboardRemove* or *telebot.types.ForceReply*) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user.
- `timeout` (`int`) – Timeout in seconds for the request.
- `data` (`str`) – function typo miss compatibility: do not use it
- `message_thread_id` (`int`) – Identifier of a message thread, in which the video will be sent

Результат
On success, the sent Message is returned.

Тип результата
telebot.types.Message

`send_video_note`(*chat_id: Union[int, str]*, *data: Union[Any, str]*, *duration: Optional[int] = None*, *length: Optional[int] = None*, *reply_to_message_id: Optional[int] = None*, *reply_markup: Optional[Union[*InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply*]] = None*, *disable_notification: Optional[bool] = None*, *timeout: Optional[int] = None*, *thumb: Optional[Union[str, Any]] = None*, *allow_sending_without_reply: Optional[bool] = None*, *protect_content: Optional[bool] = None*, *message_thread_id: Optional[int] = None*) → *Message*

As of v.4.0, Telegram clients support rounded square MPEG4 videos of up to 1 minute long. Use this method to send video messages. On success, the sent Message is returned.

Telegram documentation: https://core.telegram.org/bots/api#sendvideonote

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)

- **data** (**str** or *telebot.types.InputFile*) – Video note to send. Pass a file_id as String to send a video note that exists on the Telegram servers (recommended) or upload a new video using multipart/form-data. Sending video notes by a URL is currently unsupported
- **duration** (**int**) – Duration of sent video in seconds
- **length** (**int**) – Video width and height, i.e. diameter of the video message
- **reply_to_message_id** (**int**) – If the message is a reply, ID of the original message
- **reply_markup** (*telebot.types.InlineKeyboardMarkup* or *telebot.types.ReplyKeyboardMarkup* or *telebot.types.ReplyKeyboardRemove* or *telebot.types.ForceReply*) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user.
- **disable_notification** (**bool**) – Sends the message silently. Users will receive a notification with no sound.
- **timeout** (**int**) – Timeout in seconds for the request.
- **thumb** (**str** or *telebot.types.InputFile*) – Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass "attach://<file_attach_name>" if the thumbnail was uploaded using multipart/form-data under <file_attach_name>.
- **allow_sending_without_reply** (**bool**) – Pass True, if the message should be sent even if the specified replied-to message is not found
- **protect_content** (**bool**) – Protects the contents of the sent message from forwarding and saving
- **message_thread_id** (**int**) – Identifier of a message thread, in which the video note will be sent

Результат
On success, the sent Message is returned.

Тип результата
telebot.types.Message

send_voice(*chat_id: Union[int, str]*, *voice: Union[Any, str]*, *caption: Optional[str] = None*, *duration: Optional[int] = None*, *reply_to_message_id: Optional[int] = None*, *reply_markup: Optional[Union[*InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply*]] = None*, *parse_mode: Optional[str] = None*, *disable_notification: Optional[bool] = None*, *timeout: Optional[int] = None*, *caption_entities: Optional[List[*MessageEntity*]] = None*, *allow_sending_without_reply: Optional[bool] = None*, *protect_content: Optional[bool] = None*, *message_thread_id: Optional[int] = None*) → *Message*

Use this method to send audio files, if you want Telegram clients to display the file as a playable voice message. For this to work, your audio must be in an .OGG file encoded with OPUS (other formats may be sent as Audio or Document). On success, the sent Message is returned. Bots can currently send voice messages of up to 50 MB in size, this limit may be changed in the future.

Telegram documentation: https://core.telegram.org/bots/api#sendvoice

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `voice` (`str` or *telebot.types.InputFile*) – Audio file to send. Pass a file_id as String to send a file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a file from the Internet, or upload a new one using multipart/form-data.
- `caption` (`str`) – Voice message caption, 0-1024 characters after entities parsing
- `duration` (`int`) – Duration of the voice message in seconds
- `reply_to_message_id` (`int`) – If the message is a reply, ID of the original message
- `reply_markup` (*telebot.types.InlineKeyboardMarkup* or *telebot.types.ReplyKeyboardMarkup* or *telebot.types.ReplyKeyboardRemove* or *telebot.types.ForceReply*) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user.
- `parse_mode` (`str`) – Mode for parsing entities in the voice message caption. See formatting options for more details.
- `disable_notification` (`bool`) – Sends the message silently. Users will receive a notification with no sound.
- `timeout` (`int`) – Timeout in seconds for the request.
- `caption_entities` (`list` of *telebot.types.MessageEntity*) – A JSON-serialized list of special entities that appear in the caption, which can be specified instead of parse_mode
- `allow_sending_without_reply` (`bool`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- `protect_content` (`bool`) – Protects the contents of the sent message from forwarding and saving
- `message_thread_id` (`int`) – Identifier of a message thread, in which the message will be sent

Результат
On success, the sent Message is returned.

`set_chat_administrator_custom_title(`*chat_id: Union[int, str]*, *user_id: int*, *custom_title: str*`)` → bool

Use this method to set a custom title for an administrator in a supergroup promoted by the bot. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#setchatadministratorcustomtitle

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target supergroup (in the format @supergroupusername)
- `user_id` (`int`) – Unique identifier of the target user
- `custom_title` (`str`) – New custom title for the administrator; 0-16 characters, emoji are not allowed

Результат
True on success.

Тип результата
`bool`

`set_chat_description`(*chat_id: Union[int, str]*, *description: Optional[str] = None*) → bool

Use this method to change the description of a supergroup or a channel. The bot must be an administrator in the chat for this to work and must have the appropriate admin rights.

Telegram documentation: https://core.telegram.org/bots/api#setchatdescription

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `description` (`str`) – Str: New chat description, 0-255 characters

Результат
True on success.

Тип результата
`bool`

`set_chat_menu_button`(*chat_id: Optional[Union[int, str]] = None*, *menu_button: Optional[*MenuButton*] = None*) → bool

Use this method to change the bot's menu button in a private chat, or the default menu button. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#setchatmenubutton

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target private chat. If not specified, default bot's menu button will be changed.
- `menu_button` (*`telebot.types.MenuButton`*) – A JSON-serialized object for the new bot's menu button. Defaults to MenuButtonDefault

Результат
True on success.

Тип результата
`bool`

`set_chat_permissions`(*chat_id: Union[int, str]*, *permissions:* ChatPermissions) → bool

Use this method to set default chat permissions for all members. The bot must be an administrator in the group or a supergroup for this to work and must have the can_restrict_members admin rights.

Telegram documentation: https://core.telegram.org/bots/api#setchatpermissions

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target supergroup (in the format @supergroupusername)
- `permissions` (`telebot.types..ChatPermissions`) – New default chat permissions

Результат
True on success

Тип результата
`bool`

`set_chat_photo`(*chat_id: Union[int, str]*, *photo: Any*) → bool

Use this method to set a new profile photo for the chat. Photos can't be changed for private chats. The bot must be an administrator in the chat for this to work and must have the appropriate admin rights. Returns True on success. Note: In regular groups (non-supergroups), this method will only work if the 'All Members Are Admins' setting is off in the target group.

Telegram documentation: https://core.telegram.org/bots/api#setchatphoto

Параметры

- `chat_id` (`int` or `str`) – Int or Str: Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `photo` (`typing.Union[file_like, str]`) – InputFile: New chat photo, uploaded using multipart/form-data

Результат
True on success.

Тип результата
`bool`

`set_chat_sticker_set`(*chat_id: Union[int, str]*, *sticker_set_name: str*) → *StickerSet*

Use this method to set a new group sticker set for a supergroup. The bot must be an administrator in the chat for this to work and must have the appropriate administrator rights. Use the field can_set_sticker_set optionally returned in getChat requests to check if the bot can use this method. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#setchatstickerset

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target supergroup (in the format @supergroupusername)
- `sticker_set_name` (`str`) – Name of the sticker set to be set as the group sticker set

Результат
StickerSet object

Тип результата
telebot.types.StickerSet

`set_chat_title`(*chat_id: Union[int, str]*, *title: str*) → bool

Use this method to change the title of a chat. Titles can't be changed for private chats. The bot must be an administrator in the chat for this to work and must have the appropriate admin rights. Returns True on success. Note: In regular groups (non-supergroups), this method will only work if the 'All Members Are Admins' setting is off in the target group.

Telegram documentation: https://core.telegram.org/bots/api#setchattitle

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `title` (`str`) – New chat title, 1-255 characters

Результат
True on success.

Тип результата
bool

set_game_score(*user_id: Union[int, str]*, *score: int*, *force: Optional[bool] = None*, *chat_id: Optional[Union[int, str]] = None*, *message_id: Optional[int] = None*, *inline_message_id: Optional[str] = None*, *disable_edit_message: Optional[bool] = None*) → Union[*Message*, bool]

Sets the value of points in the game to a specific user.

Telegram documentation: https://core.telegram.org/bots/api#setgamescore

Параметры

- user_id (int or str) – User identifier
- score (int) – New score, must be non-negative
- force (bool) – Pass True, if the high score is allowed to decrease. This can be useful when fixing mistakes or banning cheaters
- chat_id (int or str) – Required if inline_message_id is not specified. Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- message_id (int) – Required if inline_message_id is not specified. Identifier of the sent message
- inline_message_id (str) – Required if chat_id and message_id are not specified. Identifier of the inline message
- disable_edit_message (bool) – Pass True, if the game message should not be automatically edited to include the current scoreboard

Результат
On success, if the message was sent by the bot, returns the edited Message, otherwise returns True.

Тип результата
types.Message or bool

set_my_commands(*commands: List[BotCommand]*, *scope: Optional[BotCommandScope] = None*, *language_code: Optional[str] = None*) → bool

Use this method to change the list of the bot's commands.

Telegram documentation: https://core.telegram.org/bots/api#setmycommands

Параметры

- commands (list of *telebot.types.BotCommand*) – List of BotCommand. At most 100 commands can be specified.
- scope (*telebot.types.BotCommandScope*) – The scope of users for which the commands are relevant. Defaults to BotCommandScopeDefault.
- language_code (str) – A two-letter ISO 639-1 language code. If empty, commands will be applied to all users from the given scope, for whose language there are no dedicated commands

Результат
True on success.

Тип результата
bool

set_my_default_administrator_rights(*rights: Optional[ChatAdministratorRights] = None*, *for_channels: Optional[bool] = None*) → bool

Use this method to change the default administrator rights requested by the bot when it's added as an administrator to groups or channels. These rights will be suggested to users, but they are are free to modify the list before adding the bot. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#setmydefaultadministratorrights

Параметры

- `rights` (*telebot.types.ChatAdministratorRights*) – A JSON-serialized object describing new default administrator rights. If not specified, the default administrator rights will be cleared.
- `for_channels` (`bool`) – Pass True to change the default administrator rights of the bot in channels. Otherwise, the default administrator rights of the bot for groups and supergroups will be changed.

Результат
True on success.

Тип результата
`bool`

set_state(*user_id: int*, *state: Union[int, str, State]*, *chat_id: Optional[int] = None*) → None

Sets a new state of a user.

Примечание: You should set both user id and chat id in order to set state for a user in a chat. Otherwise, if you only set user_id, chat_id will equal to user_id, this means that state will be set for the user in his private chat with a bot.

Параметры

- `user_id` (`int`) – User's identifier
- `state` (`int` or `str` or `telebot.types.State`) – new state. can be string, integer, or `telebot.types.State`
- `chat_id` (`int`) – Chat's identifier

Результат
None

set_sticker_position_in_set(*sticker: str*, *position: int*) → bool

Use this method to move a sticker in a set created by the bot to a specific position . Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#setstickerpositioninset

Параметры

- `sticker` (`str`) – File identifier of the sticker
- `position` (`int`) – New sticker position in the set, zero-based

Результат
On success, True is returned.

Тип результата
`bool`

`set_sticker_set_thumb`(*name: str*, *user_id: int*, *thumb: Optional[Union[str, Any]] = None*)

Use this method to set the thumbnail of a sticker set. Animated thumbnails can be set for animated sticker sets only. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#setstickersetthumb

Параметры

- `name` (`str`) – Sticker set name
- `user_id` (`int`) – User identifier
- `thumb` (`filelike object`) –

Результат

On success, True is returned.

Тип результата

`bool`

`set_update_listener`(*listener: Callable*)

Sets a listener function to be called when a new update is received.

Параметры

`listener` (*`Callable`*) – Listener function.

`set_webhook`(*url: Optional[str] = None*, *certificate: Optional[Union[str, Any]] = None*, *max_connections: Optional[int] = None*, *allowed_updates: Optional[List[str]] = None*, *ip_address: Optional[str] = None*, *drop_pending_updates: Optional[bool] = None*, *timeout: Optional[int] = None*, *secret_token: Optional[str] = None*) → bool

Use this method to specify a URL and receive incoming updates via an outgoing webhook. Whenever there is an update for the bot, we will send an HTTPS POST request to the specified URL, containing a JSON-serialized Update. In case of an unsuccessful request, we will give up after a reasonable amount of attempts. Returns True on success.

If you'd like to make sure that the webhook was set by you, you can specify secret data in the parameter secret_token. If specified, the request will contain a header "X-Telegram-Bot-Api-Secret-Token" with the secret token as content.

Telegram Documentation: https://core.telegram.org/bots/api#setwebhook

Параметры

- `url` (`str`, optional) – HTTPS URL to send updates to. Use an empty string to remove webhook integration, defaults to None
- `certificate` (`str`, optional) – Upload your public key certificate so that the root certificate in use can be checked, defaults to None
- `max_connections` (`int`, optional) – The maximum allowed number of simultaneous HTTPS connections to the webhook for update delivery, 1-100. Defaults to 40. Use lower values to limit the load on your bot's server, and higher values to increase your bot's throughput, defaults to None
- `allowed_updates` (`list`, optional) – A JSON-serialized list of the update types you want your bot to receive. For example, specify ["message", "edited_channel_post", "callback_query"] to only receive updates of these types. See Update for a complete list of available update types. Specify an empty list to receive all update types except chat_member (default). If not specified, the previous setting will be used.

Please note that this parameter doesn't affect updates created before the call to the setWebhook, so unwanted updates may be received for a short period of time. Defaults to None

- `ip_address` (`str`, optional) – The fixed IP address which will be used to send webhook requests instead of the IP address resolved through DNS, defaults to None
- `drop_pending_updates` (`bool`, optional) – Pass True to drop all pending updates, defaults to None
- `timeout` (`int`, optional) – Timeout of a request, defaults to None
- `secret_token` (`str`, optional) – A secret token to be sent in a header "X-Telegram-Bot-Api-Secret-Token" in every webhook request, 1-256 characters. Only characters A-Z, a-z, 0-9, _ and - are allowed. The header is useful to ensure that the request comes from a webhook set by you. Defaults to None

Результат
True on success.

Тип результата
`bool` if the request was successful.

`setup_middleware(`*middleware:* BaseMiddleware`)`

Registers class-based middleware.

Параметры
`middleware` (*`telebot.handler_backends.BaseMiddleware`*) – Subclass of *`telebot.handler_backends.BaseMiddleware`*

Результат
None

`shipping_query_handler(`*func*, ***kwargs*`)`

Handles new incoming shipping query. Only for invoices with flexible price. As a parameter to the decorator function, it passes *`telebot.types.ShippingQuery`* object.

Параметры
- `func` (`function`) – Function executed as a filter
- `kwargs` – Optional keyword arguments(custom filters)

Результат
None

`stop_bot()`

Stops bot by stopping polling and closing the worker pool.

`stop_message_live_location(`*chat_id: Optional[Union[int, str]] = None*, *message_id: Optional[int] = None*, *inline_message_id: Optional[str] = None*, *reply_markup: Optional[Union[*InlineKeyboardMarkup*,* ReplyKeyboardMarkup*,* ReplyKeyboardRemove*,* ForceReply*]] = None*, *timeout: Optional[int] = None*`) →` *Message*

Use this method to stop updating a live location message before live_period expires. On success, if the message is not an inline message, the edited Message is returned, otherwise True is returned.

Telegram documentation: https://core.telegram.org/bots/api#stopmessagelivelocation

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `message_id` (`int`) – Required if inline_message_id is not specified. Identifier of the message with live location to stop
- `inline_message_id` (`str`) – Required if chat_id and message_id are not specified. Identifier of the inline message with live location to stop
- `reply_markup` (*telebot.types.InlineKeyboardMarkup* or *telebot.types.ReplyKeyboardMarkup* or *telebot.types.ReplyKeyboardRemove* or *telebot.types.ForceReply*) – A JSON-serialized object for a new inline keyboard.
- `timeout` (`int`) – Timeout in seconds for the request.

Результат
On success, if the message is not an inline message, the edited Message is returned, otherwise True is returned.

Тип результата
telebot.types.Message or bool

`stop_poll`(*chat_id: Union[int, str]*, *message_id: int*, *reply_markup: Optional[Union[*InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply*]] = None*) → *Poll*

Use this method to stop a poll which was sent by the bot. On success, the stopped Poll is returned.

Telegram documentation: https://core.telegram.org/bots/api#stoppoll

Параметры

- `chat_id` (`int` | `str`) – Unique identifier for the target chat or username of the target channel
- `message_id` (`int`) – Identifier of the original message with the poll
- `reply_markup` (`InlineKeyboardMarkup` | `ReplyKeyboardMarkup` | `ReplyKeyboardRemove` | `ForceReply`) – A JSON-serialized object for a new message markup.

Результат
On success, the stopped Poll is returned.

Тип результата
types.Poll

`stop_polling`()

Stops polling.

Does not accept any arguments.

`unban_chat_member`(*chat_id: Union[int, str]*, *user_id: int*, *only_if_banned: Optional[bool] = False*) → bool

Use this method to unban a previously kicked user in a supergroup 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.

Telegram documentation: https://core.telegram.org/bots/api#unbanchatmember

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target group or username of the target supergroup or channel (in the format @username)
- `user_id` (`int`) – Unique identifier of the target user
- `only_if_banned` (`bool`) – Do nothing if the user is not banned

Результат
True on success

Тип результата
`bool`

`unban_chat_sender_chat`(*chat_id: Union[int, str]*, *sender_chat_id: Union[int, str]*) → bool

Use this method to unban a previously banned channel chat in a supergroup or channel. The bot must be an administrator for this to work and must have the appropriate administrator rights. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#unbanchatsenderchat

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `sender_chat_id` (`int` or `str`) – Unique identifier of the target sender chat.

Результат
True on success.

Тип результата
`bool`

`unpin_all_chat_messages`(*chat_id: Union[int, str]*) → bool

Use this method to unpin a all pinned messages in a supergroup chat. The bot must be an administrator in the chat for this to work and must have the appropriate admin rights. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#unpinallchatmessages

Параметры
`chat_id` (`int` or `str`) – Int or Str: Unique identifier for the target chat or username of the target channel (in the format @channelusername)

Результат
True on success.

Тип результата
`bool`

`unpin_all_forum_topic_messages`(*chat_id: Union[str, int]*, *message_thread_id: int*) → bool

Use this method to clear the list of pinned messages in a forum topic. The bot must be an administrator in the chat for this to work and must have the can_pin_messages administrator right in the supergroup. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#unpinallforumtopicmessages

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `message_thread_id` (`int`) – Identifier of the topic

Результат
On success, True is returned.

Тип результата
`bool`

`unpin_chat_message`(*chat_id: Union[int, str]*, *message_id: Optional[int] = None*) → bool

Use this method to unpin specific pinned message in a supergroup chat. The bot must be an administrator in the chat for this to work and must have the appropriate admin rights. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#unpinchatmessage

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `message_id` (`int`) – Int: Identifier of a message to unpin

Результат
True on success.

Тип результата
`bool`

`upload_sticker_file`(*user_id: int*, *png_sticker: Union[Any, str]*) → *File*

Use this method to upload a .png file with a sticker for later use in createNewStickerSet and addStickerToSet methods (can be used multiple times). Returns the uploaded File on success.

Telegram documentation: https://core.telegram.org/bots/api#uploadstickerfile

Параметры

- `user_id` (`int`) – User identifier of sticker set owner
- `png_sticker` (`filelike object`) – PNG image with the sticker, must be up to 512 kilobytes in size, dimensions must not exceed 512px, and either width or height must be exactly 512px.

Результат
On success, the sent file is returned.

Тип результата
telebot.types.File

`property user`: *User*

The User object representing this bot. Equivalent to bot.get_me() but the result is cached so only one API call is needed.

Результат
Bot's info.

Тип результата
telebot.types.User

custom_filters file

class telebot.custom_filters.AdvancedCustomFilter

Базовые классы: ABC

Advanced Custom Filter base class. Create child class with check() method. Accepts two parameters, returns bool: True - filter passed, False - filter failed. message: Message class text: Filter value given in handler

Child classes should have .key property.

Список 8: Example on creating an advanced custom filter.

```
class TextStartsFilter(AdvancedCustomFilter):
    # Filter to check whether message starts with some text.
    key = 'text_startswith'

    def check(self, message, text):
        return message.text.startswith(text)
```

check(*message*, *text*)

Perform a check.

key: str = None

class telebot.custom_filters.ChatFilter

Базовые классы: *AdvancedCustomFilter*

Check whether chat_id corresponds to given chat_id.

Список 9: Example on using this filter:

```
@bot.message_handler(chat_id=[99999])
# your function
```

key: str = 'chat_id'

class telebot.custom_filters.ForwardFilter

Базовые классы: *SimpleCustomFilter*

Check whether message was forwarded from channel or group.

Список 10: Example on using this filter:

```
@bot.message_handler(is_forwarded=True)
# your function
```

key: str = 'is_forwarded'

class telebot.custom_filters.IsAdminFilter(*bot*)

Базовые классы: *SimpleCustomFilter*

Check whether the user is administrator / owner of the chat.

Список 11: Example on using this filter:

```
@bot.message_handler(chat_types=['supergroup'], is_chat_admin=True)
# your function
```

key: str = 'is_chat_admin'

class telebot.custom_filters.IsDigitFilter

Базовые классы: *SimpleCustomFilter*

Filter to check whether the string is made up of only digits.

Список 12: Example on using this filter:

```
@bot.message_handler(is_digit=True)
# your function
```

key: str = 'is_digit'

class telebot.custom_filters.IsReplyFilter

Базовые классы: *SimpleCustomFilter*

Check whether message is a reply.

Список 13: Example on using this filter:

```
@bot.message_handler(is_reply=True)
# your function
```

key: str = 'is_reply'

class telebot.custom_filters.LanguageFilter

Базовые классы: *AdvancedCustomFilter*

Check users language_code.

Список 14: Example on using this filter:

```
@bot.message_handler(language_code=['ru'])
# your function
```

key: str = 'language_code'

class telebot.custom_filters.SimpleCustomFilter

Базовые классы: ABC

Simple Custom Filter base class. Create child class with check() method. Accepts only message, returns bool value, that is compared with given in handler.

Child classes should have .key property.

Список 15: Example on creating a simple custom filter.

```
class ForwardFilter(SimpleCustomFilter):
    # Check whether message was forwarded from channel or group.
    key = 'is_forwarded'
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    def check(self, message):
        return message.forward_date is not None
```

check(*message*)

Perform a check.

key: str = None

class telebot.custom_filters.StateFilter(*bot*)

Базовые классы: *AdvancedCustomFilter*

Filter to check state.

Список 16: Example on using this filter:

```
@bot.message_handler(state=1)
# your function
```

key: str = 'state'

class telebot.custom_filters.TextContainsFilter

Базовые классы: *AdvancedCustomFilter*

Filter to check Text message. key: text

Список 17: Example on using this filter:

```
# Will respond if any message.text contains word 'account'
@bot.message_handler(text_contains=['account'])
# your function
```

key: str = 'text_contains'

class telebot.custom_filters.TextFilter(*equals: Optional[str] = None*, *contains: Optional[Union[list, tuple]] = None*, *starts_with: Optional[Union[str, list, tuple]] = None*, *ends_with: Optional[Union[str, list, tuple]] = None*, *ignore_case: bool = False*)

Базовые классы: object

Advanced text filter to check (types.Message, types.CallbackQuery, types.InlineQuery, types.Poll)

example of usage is in examples/custom_filters/advanced_text_filter.py

Параметры

- equals (str) – string, True if object's text is equal to passed string
- contains (*list[str] or tuple[str]*) – list[str] or tuple[str], True if any string element of iterable is in text
- starts_with (str) – string, True if object's text starts with passed string
- ends_with (str) – string, True if object's text starts with passed string
- ignore_case (bool) – bool (default False), case insensitive

Исключение

ValueError – if incorrect value for a parameter was supplied

Результат
None

class telebot.custom_filters.TextMatchFilter

Базовые классы: *AdvancedCustomFilter*

Filter to check Text message.

Список 18: Example on using this filter:

```
@bot.message_handler(text=['account'])
# your function
```

key: str = 'text'

class telebot.custom_filters.TextStartsFilter

Базовые классы: *AdvancedCustomFilter*

Filter to check whether message starts with some text.

Список 19: Example on using this filter:

```
# Will work if message.text starts with 'sir'.
@bot.message_handler(text_startswith='sir')
# your function
```

key: str = 'text_startswith'

handler_backends file

class telebot.handler_backends.BaseMiddleware

Базовые классы: object

Base class for middleware. Your middlewares should be inherited from this class.

Set update_sensitive=True if you want to get different updates on different functions. For example, if you want to handle pre_process for message update, then you will have to create pre_process_message function, and so on. Same applies to post_process.

Примечание: If you want to use middleware, you have to set use_class_middlewares=True in your TeleBot instance.

Список 20: Example of class-based middlewares.

```
class MyMiddleware(BaseMiddleware):
    def __init__(self):
        self.update_sensitive = True
        self.update_types = ['message', 'edited_message']

    def pre_process_message(self, message, data):
        # only message update here
        pass

    def post_process_message(self, message, data, exception):
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        pass # only message update here for post_process

    def pre_process_edited_message(self, message, data):
        # only edited_message update here
        pass

    def post_process_edited_message(self, message, data, exception):
        pass # only edited_message update here for post_process
```

post_process(*message*, *data*, *exception*)

pre_process(*message*, *data*)

update_sensitive: bool = False

class telebot.handler_backends.CancelUpdate

Базовые классы: object

Class for canceling updates. Just return instance of this class in middleware to skip update. Update will skip handler and execution of post_process in middlewares.

class telebot.handler_backends.ContinueHandling

Базовые классы: object

Class for continue updates in handlers. Just return instance of this class in handlers to continue process.

Список 21: Example of using ContinueHandling

```
@bot.message_handler(commands=['start'])
def start(message):
    bot.send_message(message.chat.id, 'Hello World!')
    return ContinueHandling()

@bot.message_handler(commands=['start'])
def start2(message):
    bot.send_message(message.chat.id, 'Hello World2!')
```

class telebot.handler_backends.SkipHandler

Базовые классы: object

Class for skipping handlers. Just return instance of this class in middleware to skip handler. Update will go to post_process, but will skip execution of handler.

class telebot.handler_backends.State

Базовые классы: object

Class representing a state.

```
class MyStates(StatesGroup):
    my_state = State() # returns my_state:State string.
```

class telebot.handler_backends.StatesGroup

Базовые классы: object

Class representing common states.

```
class MyStates(StatesGroup):
    my_state = State() # returns my_state:State string.
```

Extensions

1.3.5 AsyncTeleBot

AsyncTeleBot methods

class telebot.async_telebot.AsyncTeleBot(*token: str*, *parse_mode: ~typing.Optional[str] = None*, *offset: ~typing.Optional[int] = None*, *exception_handler: ~typing.Optional[~telebot.async_telebot.ExceptionHandler] = None*, *state_storage: ~typing.Optional[~telebot.asyncio_storage.base_storage.StateStorageBase] = <telebot.asyncio_storage.memory_storage.StateMemoryStorage object>*, *disable_web_page_preview: ~typing.Optional[bool] = None*, *disable_notification: ~typing.Optional[bool] = None*, *protect_content: ~typing.Optional[bool] = None*, *allow_sending_without_reply: ~typing.Optional[bool] = None*, *colorful_logs: ~typing.Optional[bool] = False*)

Базовые классы: `object`

This is the main asynchronous class for Bot.

It allows you to add handlers for different kind of updates.

Usage:

Список 22: Using asynchronous implementation of TeleBot.

```
from telebot.async_telebot import AsyncTeleBot
bot = AsyncTeleBot('token') # get token from @BotFather
# now you can register other handlers/update listeners,
# and use bot methods.
# Remember to use async/await keywords when necessary.
```

See more examples in examples/ directory: https://github.com/eternnoir/pyTelegramBotAPI/tree/master/examples

Примечание: Install coloredlogs module to specify colorful_logs=True

Параметры

- `token` (`str`) – Token of a bot, obtained from @BotFather
- `parse_mode` (`str`, optional) – Default parse mode, defaults to None
- `offset` (`int`, optional) – Offset used in get_updates, defaults to None
- `exception_handler` (*Optional[ExceptionHandler], optional*) – Exception handler, which will handle the exception occured, defaults to None

- `state_storage` (`telebot.asyncio_storage.StateMemoryStorage`, optional) – Storage for states, defaults to StateMemoryStorage()
- `disable_web_page_preview` (`bool`, optional) – Default value for disable_web_page_preview, defaults to None
- `disable_notification` (`bool`, optional) – Default value for disable_notification, defaults to None
- `protect_content` (`bool`, optional) – Default value for protect_content, defaults to None
- `allow_sending_without_reply` (`bool`, optional) – Default value for allow_sending_without_reply, defaults to None
- `colorful_logs` (`bool`, optional) – Outputs colorful logs

`add_custom_filter(`*custom_filter: Union[*SimpleCustomFilter, AdvancedCustomFilter*]*`)`

Create custom filter.

Список 23: Example on checking the text of a message

```
class TextMatchFilter(AdvancedCustomFilter):
    key = 'text'

    async def check(self, message, text):
        return text == message.text
```

Параметры

`custom_filter` (*telebot.asyncio_filters.SimpleCustomFilter* or *telebot.asyncio_filters.AdvancedCustomFilter*) – Class with check(message) method.

Результат

None

`async add_data(`*user_id: int*, *chat_id: Optional[int] = None*, ***kwargs*`)`

Add data to states.

Параметры

- `user_id` (`int`) – User's identifier
- `chat_id` (`int`) – Chat's identifier
- `kwargs` – Data to add

Результат

None

`async add_sticker_to_set(`*user_id: int*, *name: str*, *emojis: str*, *png_sticker: Optional[Union[str, Any]] = None*, *tgs_sticker: Optional[Union[str, Any]] = None*, *webm_sticker: Optional[Union[str, Any]] = None*, *mask_position: Optional[*MaskPosition*] = None*`)` → bool

Use this method to add a new sticker to a set created by the bot. It's required to pass *png_sticker* or *tgs_sticker*. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#addstickertoset

Параметры

- `user_id` (`int`) – User identifier of created sticker set owner

- `name` (`str`) – Sticker set name
- `emojis` (`str`) – One or more emoji corresponding to the sticker
- `png_sticker` (`str` or `filelike object`) – PNG image with the sticker, must be up to 512 kilobytes in size, dimensions must not exceed 512px, and either width or height must be exactly 512px. Pass a file_id as a String to send a file that already exists on the Telegram servers, pass an HTTP URL as a String for Telegram to get a file from the Internet, or upload a new one using multipart/form-data.
- `tgs_sticker` (`str` or `filelike object`) – TGS animation with the sticker, uploaded using multipart/form-data.
- `webm_sticker` (`str` or `filelike object`) – WebM animation with the sticker, uploaded using multipart/form-data.
- `mask_position` (*telebot.types.MaskPosition*) – A JSON-serialized object for position where the mask should be placed on faces

Результат
On success, True is returned.

Тип результата
`bool`

async answer_callback_query(*callback_query_id: int*, *text: Optional[str] = None*, *show_alert: Optional[bool] = None*, *url: Optional[str] = None*, *cache_time: Optional[int] = None*) → bool

Use this method to send answers to callback queries sent from inline keyboards. The answer will be displayed to the user as a notification at the top of the chat screen or as an alert.

Telegram documentation: https://core.telegram.org/bots/api#answercallbackquery

Параметры

- `callback_query_id` (`int`) – Unique identifier for the query to be answered
- `text` (`str`) – Text of the notification. If not specified, nothing will be shown to the user, 0-200 characters
- `show_alert` (`bool`) – If True, an alert will be shown by the client instead of a notification at the top of the chat screen. Defaults to false.
- `url` (`str`) – URL that will be opened by the user's client. If you have created a Game and accepted the conditions via @BotFather, specify the URL that opens your game - note that this will only work if the query comes from a callback_game button.
- `cache_time` – The maximum amount of time in seconds that the result of the callback query may be cached client-side. Telegram apps will support caching starting in version 3.14. Defaults to 0.

Результат
On success, True is returned.

Тип результата
`bool`

async answer_inline_query(*inline_query_id: str*, *results: List[Any]*, *cache_time: Optional[int] = None*, *is_personal: Optional[bool] = None*, *next_offset: Optional[str] = None*, *switch_pm_text: Optional[str] = None*, *switch_pm_parameter: Optional[str] = None*) → bool

Use this method to send answers to an inline query. On success, True is returned. No more than 50 results per query are allowed.

Telegram documentation: https://core.telegram.org/bots/api#answerinlinequery

Параметры

- `inline_query_id` (`str`) – Unique identifier for the answered query
- `results` (`list of types.InlineQueryResult`) – Array of results for the inline query
- `cache_time` (`int`) – The maximum amount of time in seconds that the result of the inline query may be cached on the server.
- `is_personal` (`bool`) – Pass True, if results may be cached on the server side only for the user that sent the query.
- `next_offset` (`str`) – Pass the offset that a client should send in the next query with the same text to receive more results.
- `switch_pm_parameter` (`str`) – Deep-linking parameter for the /start message sent to the bot when user presses the switch button. 1-64 characters, only A-Z, a-z, 0-9, _ and - are allowed. Example: An inline bot that sends YouTube videos can ask the user to connect the bot to their YouTube account to adapt search results accordingly. To do this, it displays a „Connect your YouTube account“ button above the results, or even before showing any. The user presses the button, switches to a private chat with the bot and, in doing so, passes a start parameter that instructs the bot to return an OAuth link. Once done, the bot can offer a switch_inline button so that the user can easily return to the chat where they wanted to use the bot's inline capabilities.
- `switch_pm_text` (`str`) – Parameter for the start message sent to the bot when user presses the switch button

Результат
On success, True is returned.

Тип результата
`bool`

`async answer_pre_checkout_query(`*pre_checkout_query_id: int*, *ok: bool*, *error_message: Optional[str] = None*`)` → bool

Once the user has confirmed their payment and shipping details, the Bot API sends the final confirmation in the form of an Update with the field pre_checkout_query. Use this method to respond to such pre-checkout queries. On success, True is returned.

Примечание: The Bot API must receive an answer within 10 seconds after the pre-checkout query was sent.

Telegram documentation: https://core.telegram.org/bots/api#answerprecheckoutquery

Параметры

- `pre_checkout_query_id` (`int`) – Unique identifier for the query to be answered
- `ok` (`bool`) – Specify True if everything is alright (goods are available, etc.) and the bot is ready to proceed with the order. Use False if there are any problems.

- `error_message` (`str`) – Required if ok is False. Error message in human readable form that explains the reason for failure to proceed with the checkout (e.g. «Sorry, somebody just bought the last of our amazing black T-shirts while you were busy filling out your payment details. Please choose a different color or garment!»). Telegram will display this message to the user.

Результат
On success, True is returned.

Тип результата
`bool`

async answer_shipping_query(*shipping_query_id: str*, *ok: bool*, *shipping_options: Optional[List[ShippingOption]] = None*, *error_message: Optional[str] = None*) → bool

Asks for an answer to a shipping question.

Telegram documentation: https://core.telegram.org/bots/api#answershippingquery

Параметры

- `shipping_query_id` (`str`) – Unique identifier for the query to be answered
- `ok` (`bool`) – Specify True if delivery to the specified address is possible and False if there are any problems (for example, if delivery to the specified address is not possible)
- `shipping_options` (`list` of `ShippingOption`) – Required if ok is True. A JSON-serialized array of available shipping options.
- `error_message` (`str`) – Required if ok is False. Error message in human readable form that explains why it is impossible to complete the order (e.g. «Sorry, delivery to your desired address is unavailable“). Telegram will display this message to the user.

Результат
On success, True is returned.

Тип результата
`bool`

async answer_web_app_query(*web_app_query_id: str*, *result:* InlineQueryResultBase) → *SentWebAppMessage*

Use this method to set the result of an interaction with a Web App and send a corresponding message on behalf of the user to the chat from which the query originated. On success, a SentWebAppMessage object is returned.

Telegram Documentation: https://core.telegram.org/bots/api#answerwebappquery

Параметры

- `web_app_query_id` (`str`) – Unique identifier for the query to be answered
- `result` (*`telebot.types.InlineQueryResultBase`*) – A JSON-serialized object describing the message to be sent

Результат
On success, a SentWebAppMessage object is returned.

Тип результата
`telebot.types.SentWebAppMessage`

async approve_chat_join_request(*chat_id: Union[str, int]*, *user_id: Union[int, str]*) → bool

Use this method to approve a chat join request. The bot must be an administrator in the chat for this to work and must have the can_invite_users administrator right. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#approvechatjoinrequest

Параметры

- chat_id (int or str) – Unique identifier for the target chat or username of the target supergroup (in the format @supergroupusername)
- user_id (int or str) – Unique identifier of the target user

Результат

True on success.

Тип результата

bool

async ban_chat_member(*chat_id: Union[int, str]*, *user_id: int*, *until_date: Optional[Union[int, datetime]] = None*, *revoke_messages: Optional[bool] = None*) → bool

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

Telegram documentation: https://core.telegram.org/bots/api#banchatmember

Параметры

- chat_id (int or str) – Unique identifier for the target group or username of the target supergroup or channel (in the format @channelusername)
- user_id (int) – Unique identifier of the target user
- until_date (int or datetime) – Date when the user will be unbanned, unix time. If user is banned for more than 366 days or less than 30 seconds from the current time they are considered to be banned forever
- revoke_messages (bool) – Bool: Pass True to delete all messages from the chat for the user that is being removed. If False, the user will be able to see messages in the group that were sent before the user was removed. Always True for supergroups and channels.

Результат

Returns True on success.

Тип результата

bool

async ban_chat_sender_chat(*chat_id: Union[int, str]*, *sender_chat_id: Union[int, str]*) → bool

Use this method to ban a channel chat in a supergroup or a channel. The owner of the chat will not be able to send messages and join live streams on behalf of the chat, unless it is unbanned first. The bot must be an administrator in the supergroup or channel for this to work and must have the appropriate administrator rights. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#banchatsenderchat

Параметры

- chat_id (int or str) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- sender_chat_id (int or str) – Unique identifier of the target sender chat

Результат
True on success.

Тип результата
`bool`

`callback_query_handler(`*func*, ***kwargs*`)`

Handles new incoming callback query. As a parameter to the decorator function, it passes *telebot.types.CallbackQuery* object.

Параметры

- `func` (`function`) – Function executed as a filter
- `kwargs` – Optional keyword arguments(custom filters)

Результат
None

`channel_post_handler(`*commands=None*, *regexp=None*, *func=None*, *content_types=None*, ***kwargs*`)`

Handles new incoming channel post of any kind - text, photo, sticker, etc. As a parameter to the decorator function, it passes *telebot.types.Message* object.

Параметры

- `commands` (`list` of `str`) – Optional list of strings (commands to handle).
- `regexp` (`str`) – Optional regular expression.
- `func` (`function`) – Function executed as a filter
- `content_types` (`list` of `str`) – Supported message content types. Must be a list. Defaults to [„text“].
- `kwargs` – Optional keyword arguments(custom filters)

Результат
None

`chat_join_request_handler(`*func=None*, ***kwargs*`)`

Handles a request to join the chat has been sent. The bot must have the can_invite_users administrator right in the chat to receive these updates. As a parameter to the decorator function, it passes *telebot.types.ChatJoinRequest* object.

Параметры

- `func` (`function`) – Function executed as a filter
- `kwargs` – Optional keyword arguments(custom filters)

Результат
None

`chat_member_handler(`*func=None*, ***kwargs*`)`

Handles update in a status of a user in a chat. The bot must be an administrator in the chat and must explicitly specify “chat_member” in the list of allowed_updates to receive these updates. As a parameter to the decorator function, it passes *telebot.types.ChatMemberUpdated* object.

Параметры

- `func` (`function`) – Function executed as a filter
- `kwargs` – Optional keyword arguments(custom filters)

Результат
None

`chosen_inline_handler(`*func*, ***kwargs*`)`

The result of an inline query that was chosen by a user and sent to their chat partner. Please see our documentation on the feedback collecting for details on how to enable these updates for your bot. As a parameter to the decorator function, it passes *telebot.types.ChosenInlineResult* object.

Параметры

- `func` (`function`) – Function executed as a filter
- `kwargs` – Optional keyword arguments(custom filters)

Результат
None

`async close()` → bool

Use this method to close the bot instance before moving it from one local server to another. You need to delete the webhook before calling this method to ensure that the bot isn't launched again after server restart. The method will return error 429 in the first 10 minutes after the bot is launched. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#close

Результат
`bool`

`async close_forum_topic(`*chat_id: Union[str, int]*, *message_thread_id: int*`)` → bool

Use this method to close an open topic in a forum supergroup chat. The bot must be an administrator in the chat for this to work and must have the can_manage_topics administrator rights, unless it is the creator of the topic. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#closeforumtopic

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `message_thread_id` (`int`) – Identifier of the topic to close

Результат
On success, True is returned.

Тип результата
`bool`

`async close_session()`

Closes existing session of aiohttp. Use this function if you stop polling/webhooks.

`async copy_message(`*chat_id: Union[int, str]*, *from_chat_id: Union[int, str]*, *message_id: int*, *caption: Optional[str] = None*, *parse_mode: Optional[str] = None*, *caption_entities: Optional[List[MessageEntity]] = None*, *disable_notification: Optional[bool] = None*, *protect_content: Optional[bool] = None*, *reply_to_message_id: Optional[int] = None*, *allow_sending_without_reply: Optional[bool] = None*, *reply_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None*, *timeout: Optional[int] = None*, *message_thread_id: Optional[int] = None*`)` → *MessageID*

Use this method to copy messages of any kind.

Telegram documentation: https://core.telegram.org/bots/api#copymessage

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `from_chat_id` (`int` or `str`) – Unique identifier for the chat where the original message was sent (or channel username in the format @channelusername)
- `message_id` (`int`) – Message identifier in the chat specified in from_chat_id
- `caption` (`str`) – New caption for media, 0-1024 characters after entities parsing. If not specified, the original caption is kept
- `parse_mode` (`str`) – Mode for parsing entities in the new caption.
- `caption_entities` (Array of *telebot.types.MessageEntity*) – A JSON-serialized list of special entities that appear in the new caption, which can be specified instead of parse_mode
- `disable_notification` (`bool`) – Sends the message silently. Users will receive a notification with no sound.
- `protect_content` (`bool`) – Protects the contents of the sent message from forwarding and saving
- `reply_to_message_id` (`int`) – If the message is a reply, ID of the original message
- `allow_sending_without_reply` (`bool`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- `reply_markup` (*telebot.types.InlineKeyboardMarkup* or *telebot.types.ReplyKeyboardMarkup* or *telebot.types.ReplyKeyboardRemove* or *telebot.types.ForceReply*) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user.
- `timeout` (`int`) – Timeout in seconds for the request.
- `message_thread_id` (`int`) – Identifier of a message thread, in which the message will be sent

Результат

On success, the sent Message is returned.

Тип результата

telebot.types.Message

`async create_chat_invite_link(`*chat_id: Union[int, str]*, *name: Optional[str] = None*, *expire_date: Optional[Union[int, datetime]] = None*, *member_limit: Optional[int] = None*, *creates_join_request: Optional[bool] = None*`)` → *ChatInviteLink*

Use this method to create an additional invite link for a chat. The bot must be an administrator in the chat for this to work and must have the appropriate administrator rights. The link can be revoked using the method revokeChatInviteLink. Returns the new invite link as ChatInviteLink object.

Telegram documentation: https://core.telegram.org/bots/api#createchatinvitelink

Параметры

- `chat_id` (`int` or `str`) – Id: Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `name` (`str`) – Invite link name; 0-32 characters
- `expire_date` (`int` or `datetime`) – Point in time (Unix timestamp) when the link will expire
- `member_limit` (`int`) – Maximum number of users that can be members of the chat simultaneously
- `creates_join_request` (`bool`) – True, if users joining the chat via the link need to be approved by chat administrators. If True, member_limit can't be specified

Результат
Returns the new invite link as ChatInviteLink object.

Тип результата
telebot.types.ChatInviteLink

async create_forum_topic(*chat_id: int*, *name: str*, *icon_color: Optional[int] = None*, *icon_custom_emoji_id: Optional[str] = None*) → *ForumTopic*

Use this method to create a topic in a forum supergroup chat. The bot must be an administrator in the chat for this to work and must have the can_manage_topics administrator rights. Returns information about the created topic as a ForumTopic object.

Telegram documentation: https://core.telegram.org/bots/api#createforumtopic

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `name` (`str`) – Name of the topic, 1-128 characters
- `icon_color` (`int`) – Color of the topic icon in RGB format. Currently, must be one of 0x6FB9F0, 0xFFD67E, 0xCB86DB, 0x8EEE98, 0xFF93B2, or 0xFB6F5F
- `icon_custom_emoji_id` (`str`) – Custom emoji for the topic icon. Must be an emoji of type "tgs" and must be exactly 1 character long

Результат
On success, information about the created topic is returned as a ForumTopic object.

Тип результата
telebot.types.ForumTopic

async create_invoice_link(*title: str*, *description: str*, *payload: str*, *provider_token: str*, *currency: str*, *prices: List[LabeledPrice]*, *max_tip_amount: Optional[int] = None*, *suggested_tip_amounts: Optional[List[int]] = None*, *provider_data: Optional[str] = None*, *photo_url: Optional[str] = None*, *photo_size: Optional[int] = None*, *photo_width: Optional[int] = None*, *photo_height: Optional[int] = None*, *need_name: Optional[bool] = None*, *need_phone_number: Optional[bool] = None*, *need_email: Optional[bool] = None*, *need_shipping_address: Optional[bool] = None*, *send_phone_number_to_provider: Optional[bool] = None*, *send_email_to_provider: Optional[bool] = None*, *is_flexible: Optional[bool] = None*) → str

Use this method to create a link for an invoice. Returns the created invoice link as String on success.

Telegram documentation: https://core.telegram.org/bots/api#createinvoicelink

Параметры

- `title` (`str`) – Product name, 1-32 characters
- `description` (`str`) – Product description, 1-255 characters
- `payload` (`str`) – Bot-defined invoice payload, 1-128 bytes. This will not be displayed to the user, use for your internal processes.
- `provider_token` (`str`) – Payments provider token, obtained via @Botfather
- `currency` (`str`) – Three-letter ISO 4217 currency code, see https://core.telegram.org/bots/payments#supported-currencies
- `prices` (`list` of `types.LabeledPrice`) – Price breakdown, a list of components (e.g. product price, tax, discount, delivery cost, delivery tax, bonus, etc.)
- `max_tip_amount` (`int`) – The maximum accepted amount for tips in the smallest units of the currency
- `suggested_tip_amounts` (`list` of `int`) – A JSON-serialized array of suggested amounts of tips in the smallest units of the currency. At most 4 suggested tip amounts can be specified. The suggested tip amounts must be positive, passed in a strictly increased order and must not exceed max_tip_amount.
- `provider_data` (`str`) – A JSON-serialized data about the invoice, which will be shared with the payment provider. A detailed description of required fields should be provided by the payment provider.
- `photo_url` (`str`) – URL of the product photo for the invoice. Can be a photo of the goods or a photo of the invoice. People like it better when they see a photo of what they are paying for.
- `photo_size` (`int`) – Photo size in bytes
- `photo_width` (`int`) – Photo width
- `photo_height` (`int`) – Photo height
- `need_name` (`bool`) – Pass True, if you require the user's full name to complete the order
- `need_phone_number` (`bool`) – Pass True, if you require the user's phone number to complete the order
- `need_email` (`bool`) – Pass True, if you require the user's email to complete the order
- `need_shipping_address` (`bool`) – Pass True, if you require the user's shipping address to complete the order
- `send_phone_number_to_provider` (`bool`) – Pass True, if user's phone number should be sent to provider
- `send_email_to_provider` (`bool`) – Pass True, if user's email address should be sent to provider
- `is_flexible` (`bool`) – Pass True, if the final price depends on the shipping method

Результат

Created invoice link as String on success.

Тип результата

`str`

async create_new_sticker_set(*user_id: int*, *name: str*, *title: str*, *emojis: str*, *png_sticker: Optional[Union[str, Any]] = None*, *tgs_sticker: Optional[Union[str, Any]] = None*, *webm_sticker: Optional[Union[str, Any]] = None*, *contains_masks: Optional[bool] = None*, *sticker_type: Optional[str] = None*, *mask_position: Optional[MaskPosition] = None*) → bool

Use this method to create new sticker set owned by a user. The bot will be able to edit the created sticker set. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#createnewstickerset

Параметры

- `user_id` (`int`) – User identifier of created sticker set owner
- `name` (`str`) – Short name of sticker set, to be used in t.me/addstickers/ URLs (e.g., animals). Can contain only English letters, digits and underscores. Must begin with a letter, can't contain consecutive underscores and must end in «_by_<bot_username>». <bot_username> is case insensitive. 1-64 characters.
- `title` (`str`) – Sticker set title, 1-64 characters
- `emojis` (`str`) – One or more emoji corresponding to the sticker
- `png_sticker` (`str`) – PNG image with the sticker, must be up to 512 kilobytes in size, dimensions must not exceed 512px, and either width or height must be exactly 512px. Pass a file_id as a String to send a file that already exists on the Telegram servers, pass an HTTP URL as a String for Telegram to get a file from the Internet, or upload a new one using multipart/form-data.
- `tgs_sticker` (`str`) – TGS animation with the sticker, uploaded using multipart/form-data.
- `webm_sticker` (`str`) – WebM animation with the sticker, uploaded using multipart/form-data.
- `contains_masks` (`bool`) – Pass True, if a set of mask stickers should be created. Deprecated since Bot API 6.2, use sticker_type instead.
- `sticker_type` (`str`) – Optional, Type of stickers in the set, pass "regular" or "mask". Custom emoji sticker sets can't be created via the Bot API at the moment. By default, a regular sticker set is created.
- `mask_position` (*`telebot.types.MaskPosition`*) – A JSON-serialized object for position where the mask should be placed on faces

Результат

On success, True is returned.

Тип результата

`bool`

async decline_chat_join_request(*chat_id: Union[str, int]*, *user_id: Union[int, str]*) → bool

Use this method to decline a chat join request. The bot must be an administrator in the chat for this to work and must have the can_invite_users administrator right. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#declinechatjoinrequest

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target supergroup (in the format @supergroupusername)

- `user_id` (`int` or `str`) – Unique identifier of the target user

Результат
True on success.

Тип результата
`bool`

`async delete_chat_photo(`*chat_id: Union[int, str]*`)` → bool

Use this method to delete a chat photo. Photos can't be changed for private chats. The bot must be an administrator in the chat for this to work and must have the appropriate admin rights. Returns True on success. Note: In regular groups (non-supergroups), this method will only work if the 'All Members Are Admins' setting is off in the target group.

Telegram documentation: https://core.telegram.org/bots/api#deletechatphoto

Параметры
`chat_id` (`int` or `str`) – Int or Str: Unique identifier for the target chat or username of the target channel (in the format @channelusername)

Результат
True on success.

Тип результата
`bool`

`async delete_chat_sticker_set(`*chat_id: Union[int, str]*`)` → bool

Use this method to delete a group sticker set from a supergroup. The bot must be an administrator in the chat for this to work and must have the appropriate admin rights. Use the field can_set_sticker_set optionally returned in getChat requests to check if the bot can use this method. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#deletechatstickerset

Параметры
`chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target supergroup (in the format @supergroupusername)

Результат
Returns True on success.

Тип результата
`bool`

`async delete_forum_topic(`*chat_id: Union[str, int]*, *message_thread_id: int*`)` → bool

Use this method to delete a topic in a forum supergroup chat. The bot must be an administrator in the chat for this to work and must have the can_manage_topics administrator rights, unless it is the creator of the topic. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#deleteforumtopic

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `message_thread_id` (`int`) – Identifier of the topic to delete

Результат
On success, True is returned.

Тип результата
`bool`

async delete_message(*chat_id: Union[int, str]*, *message_id: int*, *timeout: Optional[int] = None*) → bool

Use this method to delete a message, including service messages, with the following limitations: - A message can only be deleted if it was sent less than 48 hours ago. - A dice message in a private chat can only be deleted if it was sent more than 24 hours ago. - Bots can delete outgoing messages in private chats, groups, and supergroups. - Bots can delete incoming messages in private chats. - Bots granted can_post_messages permissions can delete outgoing messages in channels. - If the bot is an administrator of a group, it can delete any message there. - If the bot has can_delete_messages permission in a supergroup or a channel, it can delete any message there. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#deletemessage

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `message_id` (`int`) – Identifier of the message to delete
- `timeout` (`int`) – Timeout in seconds for the request.

Результат
Returns True on success.

Тип результата
`bool`

async delete_my_commands(*scope: Optional[BotCommandScope] = None*, *language_code: Optional[int] = None*) → bool

Use this method to delete the list of the bot's commands for the given scope and user language. After deletion, higher level commands will be shown to affected users. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#deletemycommands

Параметры

- `scope` (*telebot.types.BotCommandScope*) – The scope of users for which the commands are relevant. Defaults to BotCommandScopeDefault.
- `language_code` (`str`) – A two-letter ISO 639-1 language code. If empty, commands will be applied to all users from the given scope, for whose language there are no dedicated commands

Результат
True on success.

Тип результата
`bool`

async delete_state(*user_id: int*, *chat_id: Optional[int] = None*)

Delete the current state of a user.

Параметры

- `user_id` (`int`) – User's identifier
- `chat_id` (`int`) – Chat's identifier

Результат
None

async delete_sticker_from_set(*sticker: str*) → bool

Use this method to delete a sticker from a set created by the bot. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#deletestickerfromset

Параметры
`sticker` – File identifier of the sticker

Результат
On success, True is returned.

Тип результата
`bool`

async delete_webhook(*drop_pending_updates: Optional[bool] = None*, *timeout: Optional[int] = None*) → bool

Use this method to remove webhook integration if you decide to switch back to getUpdates. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#deletewebhook

Параметры

- `drop_pending_updates` – Pass True to drop all pending updates, defaults to None
- `timeout` (`int`, optional) – Request connection timeout, defaults to None

Результат
Returns True on success.

Тип результата
`bool`

async download_file(*file_path: Optional[str]*) → bytes

Downloads file.

Параметры
`file_path` (*str*) – Path where the file should be downloaded.

Результат
bytes

Тип результата
`bytes`

async edit_chat_invite_link(*chat_id: Union[int, str]*, *invite_link: Optional[str] = None*, *name: Optional[str] = None*, *expire_date: Optional[Union[int, datetime]] = None*, *member_limit: Optional[int] = None*, *creates_join_request: Optional[bool] = None*) → *ChatInviteLink*

Use this method to edit a non-primary invite link created by the bot. The bot must be an administrator in the chat for this to work and must have the appropriate admin rights.

Telegram documentation: https://core.telegram.org/bots/api#editchatinvitelink

Параметры

- `chat_id` (`int` or `str`) – Id: Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `name` (`str`) – Invite link name; 0-32 characters
- `invite_link` (`str`) – The invite link to edit

- `expire_date` (`int` or `datetime`) – Point in time (Unix timestamp) when the link will expire
- `member_limit` (`int`) – Maximum number of users that can be members of the chat simultaneously
- `creates_join_request` (`bool`) – True, if users joining the chat via the link need to be approved by chat administrators. If True, member_limit can't be specified

Результат
Returns the new invite link as ChatInviteLink object.

Тип результата
telebot.types.ChatInviteLink

async edit_forum_topic(*chat_id: Union[int, str]*, *message_thread_id: int*, *name: str*, *icon_custom_emoji_id: str*) → bool

Use this method to edit name and icon of a topic in a forum supergroup chat. The bot must be an administrator in the chat for this to work and must have can_manage_topics administrator rights, unless it is the creator of the topic. Returns True on success.

Telegram Documentation: https://core.telegram.org/bots/api#editforumtopic

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `message_thread_id` (`int`) – Identifier of the topic to edit
- `name` (`str`) – New name of the topic, 1-128 characters
- `icon_custom_emoji_id` (`str`) – New custom emoji for the topic icon. Must be an emoji of type "tgs" and must be exactly 1 character long

Результат
On success, True is returned.

Тип результата
`bool`

async edit_message_caption(*caption: str*, *chat_id: Optional[Union[int, str]] = None*, *message_id: Optional[int] = None*, *inline_message_id: Optional[str] = None*, *parse_mode: Optional[str] = None*, *caption_entities: Optional[List[MessageEntity]] = None*, *reply_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None*) → Union[*Message*, bool]

Use this method to edit captions of messages.

Telegram documentation: https://core.telegram.org/bots/api#editmessagecaption

Параметры

- `caption` (`str`) – New caption of the message
- `chat_id` (`int | str`) – Required if inline_message_id is not specified. Unique identifier for the target chat or username of the target channel
- `message_id` (`int`) – Required if inline_message_id is not specified.
- `inline_message_id` (`str`) – Required if inline_message_id is not specified. Identifier of the inline message.

- `parse_mode` (`str`) – New caption of the message, 0-1024 characters after entities parsing
- `caption_entities` (`list` of `types.MessageEntity`) – A JSON-serialized array of objects that describe how the caption should be parsed.
- `reply_markup` (`InlineKeyboardMarkup`) – A JSON-serialized object for an inline keyboard.

Результат

On success, if edited message is sent by the bot, the edited Message is returned, otherwise True is returned.

Тип результата

`types.Message` | `bool`

async edit_message_live_location(*latitude: float*, *longitude: float*, *chat_id: Optional[Union[int, str]] = None*, *message_id: Optional[int] = None*, *inline_message_id: Optional[str] = None*, *reply_markup: Optional[Union[*InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply*]] = None*, *timeout: Optional[int] = None*, *horizontal_accuracy: Optional[float] = None*, *heading: Optional[int] = None*, *proximity_alert_radius: Optional[int] = None*) → *Message*

Use this method to edit live location messages. A location can be edited until its live_period expires or editing is explicitly

disabled by a call to stopMessageLiveLocation. On success, if the edited message is not an inline message, the edited Message is returned, otherwise True is returned.

Telegram documentation: https://core.telegram.org/bots/api#editmessagelivelocation

Параметры

- `latitude` (`float`) – Latitude of new location
- `longitude` (`float`) – Longitude of new location
- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `message_id` (`int`) – Required if inline_message_id is not specified. Identifier of the message to edit
- `reply_markup` (*telebot.types.InlineKeyboardMarkup* or *telebot.types.ReplyKeyboardMarkup* or *telebot.types.ReplyKeyboardRemove* or *telebot.types.ForceReply*) – A JSON-serialized object for a new inline keyboard.
- `timeout` (`int`) – Timeout in seconds for the request.
- `inline_message_id` (`str`) – Required if chat_id and message_id are not specified. Identifier of the inline message
- `horizontal_accuracy` (`float`) – The radius of uncertainty for the location, measured in meters; 0-1500
- `heading` (`int`) – Direction in which the user is moving, in degrees. Must be between 1 and 360 if specified.
- `proximity_alert_radius` (`int`) – The maximum distance for proximity alerts about approaching another chat member, in meters. Must be between 1 and 100000 if specified.

Результат
On success, if the edited message is not an inline message, the edited Message is returned, otherwise True is returned.

Тип результата
telebot.types.Message or bool

async edit_message_media(*media: Any, chat_id: Optional[Union[int, str]] = None, message_id: Optional[int] = None, inline_message_id: Optional[str] = None, reply_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None*) → Union[*Message*, bool]

Use this method to edit animation, audio, document, photo, or video messages. If a message is a part of a message album, then it can be edited only to a photo or a video. Otherwise, message type can be changed arbitrarily. When inline message is edited, new file can't be uploaded. Use previously uploaded file via its file_id or specify a URL.

Telegram documentation: https://core.telegram.org/bots/api#editmessagemedia

Параметры

- `media` (`InputMedia`) – A JSON-serialized object for a new media content of the message
- `chat_id` (`int` or `str`) – Required if inline_message_id is not specified. Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `message_id` (`int`) – Required if inline_message_id is not specified. Identifier of the sent message
- `inline_message_id` (`str`) – Required if chat_id and message_id are not specified. Identifier of the inline message
- `reply_markup` (*telebot.types.InlineKeyboardMarkup* or `ReplyKeyboardMarkup` or `ReplyKeyboardRemove` or `ForceReply`) – A JSON-serialized object for an inline keyboard.

Результат
On success, if edited message is sent by the bot, the edited Message is returned, otherwise True is returned.

Тип результата
`types.Message` or `bool`

async edit_message_reply_markup(*chat_id: Optional[Union[int, str]] = None, message_id: Optional[int] = None, inline_message_id: Optional[str] = None, reply_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None*) → Union[*Message*, bool]

Use this method to edit only the reply markup of messages.

Telegram documentation: https://core.telegram.org/bots/api#editmessagereplymarkup

Параметры

- `chat_id` (`int` or `str`) – Required if inline_message_id is not specified. Unique identifier for the target chat or username of the target channel (in the format @channelusername)

- `message_id` (`int`) – Required if inline_message_id is not specified. Identifier of the sent message
- `inline_message_id` (`str`) – Required if chat_id and message_id are not specified. Identifier of the inline message
- `reply_markup` (`InlineKeyboardMarkup` or `ReplyKeyboardMarkup` or `ReplyKeyboardRemove` or `ForceReply`) – A JSON-serialized object for an inline keyboard.

Результат
On success, if edited message is sent by the bot, the edited Message is returned, otherwise True is returned.

Тип результата
`types.Message` or `bool`

`async edit_message_text(`*text: str*, *chat_id: Optional[Union[int, str]] = None*, *message_id: Optional[int] = None*, *inline_message_id: Optional[str] = None*, *parse_mode: Optional[str] = None*, *entities: Optional[List[*MessageEntity*]] = None*, *disable_web_page_preview: Optional[bool] = None*, *reply_markup: Optional[Union[*InlineKeyboardMarkup*,* ReplyKeyboardMarkup*,* ReplyKeyboardRemove*,* ForceReply*]] = None*`)` → Union[*Message*, bool]

Use this method to edit text and game messages.

Telegram documentation: https://core.telegram.org/bots/api#editmessagetext

Параметры

- `text` (`str`) – New text of the message, 1-4096 characters after entities parsing
- `chat_id` (`int` or `str`) – Required if inline_message_id is not specified. Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `message_id` (`int`) – Required if inline_message_id is not specified. Identifier of the sent message
- `inline_message_id` (`str`) – Required if chat_id and message_id are not specified. Identifier of the inline message
- `parse_mode` (`str`) – Mode for parsing entities in the message text.
- `entities` (List of *`telebot.types.MessageEntity`*) – List of special entities that appear in the message text, which can be specified instead of parse_mode
- `disable_web_page_preview` (`bool`) – Disables link previews for links in this message
- `reply_markup` (`InlineKeyboardMarkup`) – A JSON-serialized object for an inline keyboard.

Результат
On success, if edited message is sent by the bot, the edited Message is returned, otherwise True is returned.

Тип результата
`types.Message` or `bool`

`edited_channel_post_handler(`*commands=None*, *regexp=None*, *func=None*, *content_types=None*, ***kwargs*`)`

Handles new version of a channel post that is known to the bot and was edited. As a parameter to the decorator function, it passes *telebot.types.Message* object.

Параметры

- `commands` (`list` of `str`) – Optional list of strings (commands to handle).
- `regexp` (`str`) – Optional regular expression.
- `func` (`function`) – Function executed as a filter
- `content_types` (`list` of `str`) – Supported message content types. Must be a list. Defaults to [„text“].
- `kwargs` – Optional keyword arguments(custom filters)

Результат

`edited_message_handler(`*commands=None*, *regexp=None*, *func=None*, *content_types=None*, *chat_types=None*, ***kwargs*`)`

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

As a parameter to the decorator function, it passes *telebot.types.Message* object.

Параметры

- `commands` (`list` of `str`) – Optional list of strings (commands to handle).
- `regexp` (`str`) – Optional regular expression.
- `func` (`function`) – Function executed as a filter
- `content_types` (`list` of `str`) – Supported message content types. Must be a list. Defaults to [„text“].
- `chat_types` (`list` of `str`) – list of chat types
- `kwargs` – Optional keyword arguments(custom filters)

Результат

None

`enable_saving_states(`*filename='./.state-save/states.pkl'*`)`

Enable saving states (by default saving disabled)

Примечание: It is recommended to pass a `StatePickleStorage` instance as state_storage to TeleBot class.

Параметры

`filename` (`str`, optional) – Filename of saving file, defaults to «./.state-save/states.pkl»

`async export_chat_invite_link(`*chat_id: Union[int, str]*`)` → str

Use this method to export an invite link to a supergroup or a channel. The bot must be an administrator in the chat for this to work and must have the appropriate admin rights.

Telegram documentation: https://core.telegram.org/bots/api#exportchatinvitelink

Параметры

`chat_id` (`int` or `str`) – Id: Unique identifier for the target chat or username of the target channel (in the format @channelusername)

Результат
exported invite link as String on success.

Тип результата
str

async forward_message(*chat_id: Union[int, str]*, *from_chat_id: Union[int, str]*, *message_id: int*, *disable_notification: Optional[bool] = None*, *protect_content: Optional[bool] = None*, *timeout: Optional[int] = None*, *message_thread_id: Optional[int] = None*) → *Message*

Use this method to forward messages of any kind.

Telegram documentation: https://core.telegram.org/bots/api#forwardmessage

Параметры

- disable_notification (bool) – Sends the message silently. Users will receive a notification with no sound
- chat_id (int or str) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- from_chat_id (int or str) – Unique identifier for the chat where the original message was sent (or channel username in the format @channelusername)
- message_id (int) – Message identifier in the chat specified in from_chat_id
- protect_content (bool) – Protects the contents of the forwarded message from forwarding and saving
- timeout (int) – Timeout in seconds for the request.
- message_thread_id (int) – Unique identifier for the target message thread (topic) of the forum; for forum supergroups only

Результат
On success, the sent Message is returned.

Тип результата
telebot.types.Message

async get_chat(*chat_id: Union[int, str]*) → *Chat*

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.). Returns a Chat object on success.

Telegram documentation: https://core.telegram.org/bots/api#getchat

Параметры
chat_id (int or str) – Unique identifier for the target chat or username of the target supergroup or channel (in the format @channelusername)

Результат
Chat information

Тип результата
telebot.types.Chat

async get_chat_administrators(*chat_id: Union[int, str]*) → List[*ChatMember*]

Use this method to get a list of administrators in a chat. On success, returns an Array of ChatMember objects that contains information about all chat administrators except other bots.

Telegram documentation: https://core.telegram.org/bots/api#getchatadministrators

Параметры
`chat_id` – Unique identifier for the target chat or username of the target supergroup or channel (in the format @channelusername)

Результат
List made of ChatMember objects.

Тип результата
`list` of *telebot.types.ChatMember*

`async get_chat_member(`*chat_id: Union[int, str]*, *user_id: int*`)` → *ChatMember*

Use this method to get information about a member of a chat. Returns a ChatMember object on success.

Telegram documentation: https://core.telegram.org/bots/api#getchatmember

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target supergroup (in the format @supergroupusername)
- `user_id` (`int`) – Unique identifier of the target user

Результат
Returns ChatMember object on success.

Тип результата
telebot.types.ChatMember

`async get_chat_member_count(`*chat_id: Union[int, str]*`)` → int

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

Telegram documentation: https://core.telegram.org/bots/api#getchatmembercount

Параметры
`chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target supergroup or channel (in the format @channelusername)

Результат
Number of members in the chat.

Тип результата
`int`

`get_chat_members_count(`***kwargs*`)`

`async get_chat_menu_button(`*chat_id: Optional[Union[int, str]] = None*`)` → *MenuButton*

Use this method to get the current value of the bot's menu button in a private chat, or the default menu button. Returns MenuButton on success.

Telegram Documentation: https://core.telegram.org/bots/api#getchatmenubutton

Параметры
`chat_id` (`int` or `str`) – Unique identifier for the target private chat. If not specified, default bot's menu button will be returned.

Результат
types.MenuButton

Тип результата
telebot.types.MenuButton

async get_custom_emoji_stickers(*custom_emoji_ids: List[str]*) → List[*Sticker*]

Use this method to get information about custom emoji stickers by their identifiers. Returns an Array of Sticker objects.

Параметры
custom_emoji_ids (list of str) – List of custom emoji identifiers. At most 200 custom emoji identifiers can be specified.

Результат
Returns an Array of Sticker objects.

Тип результата
list of *telebot.types.Sticker*

async get_file(*file_id: Optional[str]*) → *File*

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 20MB in size. On success, a File object is returned. It is guaranteed that the link will be valid for at least 1 hour. When the link expires, a new one can be requested by calling get_file again.

Telegram documentation: https://core.telegram.org/bots/api#getfile

Параметры
file_id (str) – File identifier

Результат
telebot.types.File

async get_file_url(*file_id: Optional[str]*) → str

Get a valid URL for downloading a file.

Параметры
file_id (str) – File identifier to get download URL for.

Результат
URL for downloading the file.

Тип результата
str

async get_forum_topic_icon_stickers() → List[*Sticker*]

Use this method to get custom emoji stickers, which can be used as a forum topic icon by any user. Requires no parameters. Returns an Array of Sticker objects.

Telegram documentation: https://core.telegram.org/bots/api#getforumtopiciconstickers

Результат
On success, a list of StickerSet objects is returned.

Тип результата
List[*telebot.types.StickerSet*]

async get_game_high_scores(*user_id: int*, *chat_id: Optional[Union[int, str]] = None*, *message_id: Optional[int] = None*, *inline_message_id: Optional[str] = None*) → List[*GameHighScore*]

Use this method to get data for high score tables. Will return the score of the specified user and several of their neighbors in a game. On success, returns an Array of GameHighScore objects.

This method will currently return scores for the target user, plus two of their closest neighbors on each side. Will also return the top three users if the user and their neighbors are not among them. Please note that this behavior is subject to change.

Telegram documentation: https://core.telegram.org/bots/api#getgamehighscores

Параметры

- `user_id` (`int`) – User identifier
- `chat_id` (`int` or `str`) – Required if inline_message_id is not specified. Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `message_id` (`int`) – Required if inline_message_id is not specified. Identifier of the sent message
- `inline_message_id` (`str`) – Required if chat_id and message_id are not specified. Identifier of the inline message

Результат

On success, returns an Array of GameHighScore objects.

Тип результата

List[*types.GameHighScore*]

`async get_me()` → *User*

Returns basic information about the bot in form of a User object.

Telegram documentation: https://core.telegram.org/bots/api#getme

`async get_my_commands(`*scope: Optional[*BotCommandScope*]*, *language_code: Optional[str]*`)` → List[*BotCommand*]

Use this method to get the current list of the bot's commands. Returns List of BotCommand on success.

Telegram documentation: https://core.telegram.org/bots/api#getmycommands

Параметры

- `scope` (*`telebot.types.BotCommandScope`*) – The scope of users for which the commands are relevant. Defaults to BotCommandScopeDefault.
- `language_code` (`str`) – A two-letter ISO 639-1 language code. If empty, commands will be applied to all users from the given scope, for whose language there are no dedicated commands

Результат

List of BotCommand on success.

Тип результата

`list` of *`telebot.types.BotCommand`*

`async get_my_default_administrator_rights(`*for_channels: Optional[bool] = None*`)` → *ChatAdministratorRights*

Use this method to get the current default administrator rights of the bot. Returns ChatAdministratorRights on success.

Telegram documentation: https://core.telegram.org/bots/api#getmydefaultadministratorrights

Параметры

`for_channels` (`bool`) – Pass True to get the default administrator rights of the bot in channels. Otherwise, the default administrator rights of the bot for groups and supergroups will be returned.

Результат

Returns ChatAdministratorRights on success.

Тип результата
telebot.types.ChatAdministratorRights

async get_state(*user_id*, *chat_id: Optional[int] = None*)

Gets current state of a user. Not recommended to use this method. But it is ok for debugging.

Параметры

- `user_id` (`int`) – User's identifier
- `chat_id` (`int`) – Chat's identifier

Результат
state of a user

Тип результата
`int` or `str` or `telebot.types.State`

async get_sticker_set(*name: str*) → *StickerSet*

Use this method to get a sticker set. On success, a StickerSet object is returned.

Telegram documentation: https://core.telegram.org/bots/api#getstickerset

Параметры
`name` (`str`) – Sticker set name

Результат
On success, a StickerSet object is returned.

Тип результата
telebot.types.StickerSet

async get_updates(*offset: Optional[int] = None*, *limit: Optional[int] = None*, *timeout: Optional[int] = 20*, *allowed_updates: Optional[List] = None*, *request_timeout: Optional[int] = None*) → List[*Update*]

Use this method to receive incoming updates using long polling (wiki). An Array of Update objects is returned.

Telegram documentation: https://core.telegram.org/bots/api#getupdates

Параметры

- `offset` (`int`, optional) – Identifier of the first update to be returned. Must be greater by one than the highest among the identifiers of previously received updates. By default, updates starting with the earliest unconfirmed update are returned. An update is considered confirmed as soon as getUpdates is called with an offset higher than its update_id. The negative offset can be specified to retrieve updates starting from -offset update from the end of the updates queue. All previous updates will forgotten.
- `limit` (`int`, optional) – Limits the number of updates to be retrieved. Values between 1-100 are accepted. Defaults to 100.
- `timeout` (`int`, optional) – Request connection timeout
- `allowed_updates` (`list`, optional) – Array of string. List the types of updates you want your bot to receive.
- `long_polling_timeout` (`int`, optional) – Timeout in seconds for long polling.

Результат
An Array of Update objects is returned.

Тип результата
list of *telebot.types.Update*

async get_user_profile_photos(*user_id: int*, *offset: Optional[int] = None*, *limit: Optional[int] = None*) → *UserProfilePhotos*

Use this method to get a list of profile pictures for a user. Returns a *telebot.types.UserProfilePhotos* object.

Telegram documentation: https://core.telegram.org/bots/api#getuserprofilephotos

Параметры

- **user_id** (int) – Unique identifier of the target user
- **offset** (int) – Sequential number of the first photo to be returned. By default, all photos are returned.
- **limit** (int) – Limits the number of photos to be retrieved. Values between 1-100 are accepted. Defaults to 100.

Результат
UserProfilePhotos

Тип результата
telebot.types.UserProfilePhotos

async get_webhook_info(*timeout: Optional[int] = None*) → *WebhookInfo*

Use this method to get current webhook status. Requires no parameters. On success, returns a WebhookInfo object. If the bot is using getUpdates, will return an object with the url field empty.

Telegram documentation: https://core.telegram.org/bots/api#getwebhookinfo

Параметры
timeout (int, optional) – Request connection timeout

Результат
On success, returns a WebhookInfo object.

Тип результата
telebot.types.WebhookInfo

async infinity_polling(*timeout: Optional[int] = 20*, *skip_pending: Optional[bool] = False*, *request_timeout: Optional[int] = None*, *logger_level: Optional[int] = 40*, *allowed_updates: Optional[List[str]] = None*, *restart_on_change: Optional[bool] = False*, *path_to_watch: Optional[str] = None*, **args*, ***kwargs*)

Wrap polling with infinite loop and exception handling to avoid bot stops polling.

Примечание: Install watchdog and psutil before using restart_on_change option.

Параметры

- **timeout** (int) – Timeout in seconds for get_updates(Defaults to None)
- **skip_pending** (bool) – skip old updates
- **request_timeout** (int) – Aiohttp's request timeout. Defaults to 5 minutes(aiohttp.ClientTimeout).
- **logger_level** (int) – Custom logging level for infinity_polling logging. Use logger levels from logging as a value. None/NOTSET = no error logging

- `allowed_updates` (`list` of `str`) – A list of the update types you want your bot to receive. For example, specify [“message”, “edited_channel_post”, “callback_query”] to only receive updates of these types. See util.update_types for a complete list of available update types. Specify an empty list to receive all update types except chat_member (default). If not specified, the previous setting will be used.

  Please note that this parameter doesn't affect updates created before the call to the get_updates, so unwanted updates may be received for a short period of time.
- `restart_on_change` (`bool`) – Restart a file on file(s) change. Defaults to False
- `path_to_watch` (`str`) – Path to watch for changes. Defaults to current directory

Результат
None

`inline_handler`(*func*, ***kwargs*)

Handles new incoming inline query. As a parameter to the decorator function, it passes *telebot.types.InlineQuery* object.

Параметры

- `func` (`function`) – Function executed as a filter
- `kwargs` – Optional keyword arguments(custom filters)

Результат
None

`async kick_chat_member`(*chat_id: Union[int, str]*, *user_id: int*, *until_date: Optional[Union[int, datetime]] = None*, *revoke_messages: Optional[bool] = None*) → bool

This function is deprecated. Use *ban_chat_member* instead

`async leave_chat`(*chat_id: Union[int, str]*) → bool

Use this method for your bot to leave a group, supergroup or channel. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#leavechat

Параметры
`chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target supergroup or channel (in the format @channelusername)

Результат
`bool`

`async log_out()` → bool

Use this method to log out from the cloud Bot API server before launching the bot locally. You MUST log out the bot before running it locally, otherwise there is no guarantee that the bot will receive updates. After a successful call, you can immediately log in on a local server, but will not be able to log in back to the cloud Bot API server for 10 minutes. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#logout

Результат
True on success.

Тип результата
`bool`

`message_handler`(*commands=None*, *regexp=None*, *func=None*, *content_types=None*, *chat_types=None*, ***kwargs*)

Handles ew incoming message of any kind - text, photo, sticker, etc. As a parameter to the decorator function, it passes *telebot.types.Message* object. All message handlers are tested in the order they were added.

Example:

Список 24: Usage of message_handler

```
bot = TeleBot('TOKEN')

# Handles all messages which text matches regexp.
@bot.message_handler(regexp='someregexp')
async def command_help(message):
    await bot.send_message(message.chat.id, 'Did someone call for help?')

# Handles messages in private chat
@bot.message_handler(chat_types=['private']) # You can add more chat types
async def command_help(message):
    await bot.send_message(message.chat.id, 'Private chat detected, sir!')

# Handle all sent documents of type 'text/plain'.
@bot.message_handler(func=lambda message: message.document.mime_type == 'text/
↪plain',
    content_types=['document'])
async def command_handle_document(message):
    await bot.send_message(message.chat.id, 'Document received, sir!')

# Handle all other messages.
@bot.message_handler(func=lambda message: True, content_types=['audio', 'photo',
↪ 'voice', 'video', 'document',
    'text', 'location', 'contact', 'sticker'])
async def default_command(message):
    await bot.send_message(message.chat.id, "This is the default command␣
↪handler.")
```

Параметры

- `commands` (`list` of `str`) – Optional list of strings (commands to handle).
- `regexp` (`str`) – Optional regular expression.
- `func` – Optional lambda function. The lambda receives the message to test as the first parameter. It must return True if the command should handle the message.
- `content_types` (`list` of `str`) – Supported message content types. Must be a list. Defaults to [„text“].
- `chat_types` (`list` of `str`) – list of chat types
- `kwargs` – Optional keyword arguments(custom filters)

Результат

decorated function

`my_chat_member_handler`(*func=None*, ***kwargs*)

Handles update in a status of a bot. For private chats, this update is received only when the

bot is blocked or unblocked by the user. As a parameter to the decorator function, it passes *telebot.types.ChatMemberUpdated* object.

Параметры

- `func` (`function`) – Function executed as a filter
- `kwargs` – Optional keyword arguments(custom filters)

Результат
None

`async pin_chat_message(`*chat_id: Union[int, str]*, *message_id: int*, *disable_notification: Optional[bool] = False*`)` → bool

Use this method to pin a message in a supergroup. The bot must be an administrator in the chat for this to work and must have the appropriate admin rights. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#pinchatmessage

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `message_id` (`int`) – Identifier of a message to pin
- `disable_notification` (`bool`) – Pass True, if it is not necessary to send a notification to all group members about the new pinned message

Результат
True on success.

Тип результата
`bool`

`poll_answer_handler(`*func=None*, ***kwargs*`)`

Handles change of user's answer in a non-anonymous poll(when user changes the vote). Bots receive new votes only in polls that were sent by the bot itself. As a parameter to the decorator function, it passes *telebot.types.PollAnswer* object.

Параметры

- `func` (`function`) – Function executed as a filter
- `kwargs` – Optional keyword arguments(custom filters)

Результат
None

`poll_handler(`*func*, ***kwargs*`)`

Handles new state of a poll. Bots receive only updates about stopped polls and polls, which are sent by the bot As a parameter to the decorator function, it passes *telebot.types.Poll* object.

Параметры

- `func` (`function`) – Function executed as a filter
- `kwargs` – Optional keyword arguments(custom filters)

Результат
None

async polling(*non_stop: bool = False*, *skip_pending=False*, *interval: int = 0*, *timeout: int = 20*, *request_timeout: Optional[int] = None*, *allowed_updates: Optional[List[str]] = None*, *none_stop: Optional[bool] = None*, *restart_on_change: Optional[bool] = False*, *path_to_watch: Optional[str] = None*)

Runs bot in long-polling mode in a main loop. This allows the bot to retrieve Updates automagically and notify listeners and message handlers accordingly.

Warning: Do not call this function more than once!

Always gets updates.

Примечание: Set non_stop=True if you want your bot to continue receiving updates if there is an error.

Примечание: Install watchdog and psutil before using restart_on_change option.

Параметры

- `non_stop` (`bool`) – Do not stop polling when an ApiException occurs.
- `skip_pending` (`bool`) – skip old updates
- `interval` (`int`) – Delay between two update retrivals
- `timeout` (`int`) – Request connection timeout
- `request_timeout` (`int`) – Timeout in seconds for get_updates(Defaults to None)
- `allowed_updates` (`list` of `str`) – A list of the update types you want your bot to receive. For example, specify [“message”, “edited_channel_post”, “callback_query”] to only receive updates of these types. See util.update_types for a complete list of available update types. Specify an empty list to receive all update types except chat_member (default). If not specified, the previous setting will be used.

  Please note that this parameter doesn't affect updates created before the call to the get_updates, so unwanted updates may be received for a short period of time.
- `none_stop` (`bool`) – Deprecated, use non_stop. Old typo, kept for backward compatibility.
- `restart_on_change` (`bool`) – Restart a file on file(s) change. Defaults to False.
- `path_to_watch` (`str`) – Path to watch for changes. Defaults to current directory

Результат

pre_checkout_query_handler(*func*, ***kwargs*)

New incoming pre-checkout query. Contains full information about checkout. As a parameter to the decorator function, it passes *telebot.types.PreCheckoutQuery* object.

Параметры

- `func` (`function`) – Function executed as a filter
- `kwargs` – Optional keyword arguments(custom filters)

Результат

None

async process_new_updates(*updates: List[Update]*)

Process new updates. Just pass list of updates - each update should be instance of Update object.

Параметры

updates (list of *telebot.types.Update*) – list of updates

Результат

None

async promote_chat_member(*chat_id: Union[int, str]*, *user_id: int*, *can_change_info: Optional[bool] = None*, *can_post_messages: Optional[bool] = None*, *can_edit_messages: Optional[bool] = None*, *can_delete_messages: Optional[bool] = None*, *can_invite_users: Optional[bool] = None*, *can_restrict_members: Optional[bool] = None*, *can_pin_messages: Optional[bool] = None*, *can_promote_members: Optional[bool] = None*, *is_anonymous: Optional[bool] = None*, *can_manage_chat: Optional[bool] = None*, *can_manage_video_chats: Optional[bool] = None*, *can_manage_voice_chats: Optional[bool] = None*, *can_manage_topics: Optional[bool] = None*) → bool

Use this method to promote or demote a user in a supergroup or a channel. The bot must be 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.

Telegram documentation: https://core.telegram.org/bots/api#promotechatmember

Параметры

- chat_id (int or str) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- user_id (int) – Unique identifier of the target user
- can_change_info (bool) – Pass True, if the administrator can change chat title, photo and other settings
- can_post_messages (bool) – Pass True, if the administrator can create channel posts, channels only
- can_edit_messages (bool) – Pass True, if the administrator can edit messages of other users, 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 administrator 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 administrator can pin messages, supergroups 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 him)
- is_anonymous (bool) – Pass True, if the administrator's presence in the chat is hidden

- `can_manage_chat` (`bool`) – Pass True, if the administrator can access the chat event log, chat statistics, message statistics in channels, see channel members, see anonymous administrators in supergroups and ignore slow mode. Implied by any other administrator privilege
- `can_manage_video_chats` (`bool`) – Pass True, if the administrator can manage voice chats For now, bots can use this privilege only for passing to other administrators.
- `can_manage_voice_chats` (`bool`) – Deprecated, use can_manage_video_chats.
- `can_manage_topics` (`bool`) – Pass True if the user is allowed to create, rename, close, and reopen forum topics, supergroups only

Результат
True on success.

Тип результата
`bool`

`register_callback_query_handler(`*callback: Awaitable*, *func: Callable*, *pass_ bot: Optional[bool] = False*, ***kwargs*`)`

Registers callback query handler.

Параметры

- `callback` (`Awaitable`) – function to be called
- `func` (`function`) – Function executed as a filter
- `pass_bot` (`bool`) – True if you need to pass TeleBot instance to handler(useful for separating handlers into different files)
- `kwargs` – Optional keyword arguments(custom filters)

Результат
None

`register_channel_post_handler(`*callback: Awaitable*, *content_ types: Optional[List[str]] = None*, *commands: Optional[List[str]] = None*, *regexp: Optional[str] = None*, *func: Optional[Callable] = None*, *pass_ bot: Optional[bool] = False*, ***kwargs*`)`

Registers channel post message handler.

Параметры

- `callback` (`Awaitable`) – function to be called
- `content_types` (`list` of `str`) – Supported message content types. Must be a list. Defaults to [„text“].
- `commands` (`list` of `str`) – list of commands
- `regexp` (`str`) – Regular expression
- `func` (`function`) – Function executed as a filter
- `pass_bot` (`bool`) – True if you need to pass TeleBot instance to handler(useful for separating handlers into different files)
- `kwargs` – Optional keyword arguments(custom filters)

Результат
None

`register_chat_join_request_handler`(*callback: Awaitable, func: Optional[Callable] = None, pass_bot: Optional[bool] = False, **kwargs*)

Registers chat join request handler.

Параметры

- `callback` (`Awaitable`) – function to be called
- `func` (`function`) – Function executed as a filter
- `pass_bot` (`bool`) – True if you need to pass TeleBot instance to handler(useful for separating handlers into different files)
- `kwargs` – Optional keyword arguments(custom filters)

Результат
None

`register_chat_member_handler`(*callback: Awaitable, func: Optional[Callable] = None, pass_bot: Optional[bool] = False, **kwargs*)

Registers chat member handler.

Параметры

- `callback` (`Awaitable`˙) – function to be called
- `func` (`function`) – Function executed as a filter
- `pass_bot` (`bool`) – True if you need to pass TeleBot instance to handler(useful for separating handlers into different files)
- `kwargs` – Optional keyword arguments(custom filters)

:return:None

`register_chosen_inline_handler`(*callback: Awaitable, func: Callable, pass_bot: Optional[bool] = False, **kwargs*)

Registers chosen inline handler.

Параметры

- `callback` (`Awaitable`) – function to be called
- `func` (`function`) – Function executed as a filter
- `pass_bot` (`bool`) – True if you need to pass TeleBot instance to handler(useful for separating handlers into different files)
- `kwargs` – Optional keyword arguments(custom filters)

Результат
None

`register_edited_channel_post_handler`(*callback: Awaitable, content_types: Optional[List[str]] = None, commands: Optional[List[str]] = None, regexp: Optional[str] = None, func: Optional[Callable] = None, pass_bot: Optional[bool] = False, **kwargs*)

Registers edited channel post message handler.

Параметры

- `callback` (`Awaitable`) – function to be called
- `content_types` (`list` of `str`) – Supported message content types. Must be a list. Defaults to [„text“].

- `commands` (`list` of `str`) – list of commands
- `regexp` (`str`) – Regular expression
- `func` (`function`) – Function executed as a filter
- `pass_bot` (`bool`) – True if you need to pass TeleBot instance to handler(useful for separating handlers into different files)
- `kwargs` – Optional keyword arguments(custom filters)

Результат
decorated function

`register_edited_message_handler(`*callback: Awaitable*, *content_types: Optional[List[str]] = None*, *commands: Optional[List[str]] = None*, *regexp: Optional[str] = None*, *func: Optional[Callable] = None*, *chat_types: Optional[List[str]] = None*, *pass_bot: Optional[bool] = False*, ***kwargs*`)`

Registers edited message handler.

Параметры

- `callback` (`Awaitable`) – function to be called
- `content_types` (`list` of `str`) – Supported message content types. Must be a list. Defaults to [„text“].
- `commands` (`list` of `str`) – list of commands
- `regexp` (`str`) – Regular expression
- `func` (`function`) – Function executed as a filter
- `chat_types` (`bool`) – True for private chat
- `pass_bot` (`bool`) – True if you need to pass TeleBot instance to handler(useful for separating handlers into different files)
- `kwargs` – Optional keyword arguments(custom filters)

Результат
None

`register_inline_handler(`*callback: Awaitable*, *func: Callable*, *pass_bot: Optional[bool] = False*, ***kwargs*`)`

Registers inline handler.

Параметры

- `callback` (`Awaitable`) – function to be called
- `func` (`function`) – Function executed as a filter
- `pass_bot` (`bool`) – True if you need to pass TeleBot instance to handler(useful for separating handlers into different files)
- `kwargs` – Optional keyword arguments(custom filters)

Результат
decorated function

`register_message_handler(`*callback: Awaitable*, *content_types: Optional[List[str]] = None*, *commands: Optional[List[str]] = None*, *regexp: Optional[str] = None*, *func: Optional[Callable] = None*, *chat_types: Optional[List[str]] = None*, *pass_bot: Optional[bool] = False*, ***kwargs*`)`

Registers message handler.

Параметры

- `callback` (`Awaitable`) – function to be called
- `content_types` (`list` of `str`) – Supported message content types. Must be a list. Defaults to [„text“].
- `commands` (`list` of `str`) – list of commands
- `regexp` (`str`) –
- `func` (`function`) – Function executed as a filter
- `chat_types` (`list` of `str`) – List of chat types
- `pass_bot` (`bool`) – True if you need to pass TeleBot instance to handler(useful for separating handlers into different files)
- `kwargs` – Optional keyword arguments(custom filters)

Результат
None

`register_my_chat_member_handler(`*callback: Awaitable*, *func: Optional[Callable] = None*, *pass_ bot: Optional[bool] = False*, ***kwargs*`)`

Registers my chat member handler.

Параметры

- `callback` (`Awaitable`) – function to be called
- `func` (`function`) – Function executed as a filter
- `pass_bot` (`bool`) – True if you need to pass TeleBot instance to handler(useful for separating handlers into different files)
- `kwargs` – Optional keyword arguments(custom filters)

Результат
None

`register_poll_answer_handler(`*callback: Awaitable*, *func: Callable*, *pass_ bot: Optional[bool] = False*, ***kwargs*`)`

Registers poll answer handler.

Параметры

- `callback` (`Awaitable`) – function to be called
- `func` (`function`) – Function executed as a filter
- `pass_bot` (`bool`) – True if you need to pass TeleBot instance to handler(useful for separating handlers into different files)
- `kwargs` – Optional keyword arguments(custom filters)

Результат
None

`register_poll_handler(`*callback: Awaitable*, *func: Callable*, *pass_ bot: Optional[bool] = False*, ***kwargs*`)`

Registers poll handler.

Параметры

- `callback` (`Awaitable`) – function to be called
- `func` (`function`) – Function executed as a filter
- `pass_bot` (`bool`) – True if you need to pass TeleBot instance to handler(useful for separating handlers into different files)
- `kwargs` – Optional keyword arguments(custom filters)

Результат
None

`register_pre_checkout_query_handler(`*callback: Awaitable*, *func: Callable*, *pass_ bot: Optional[bool] = False*, ***kwargs*`)`

Registers pre-checkout request handler.

Параметры

- `callback` (`Awaitable`) – function to be called
- `func` – Function executed as a filter
- `pass_bot` (`bool`) – True if you need to pass TeleBot instance to handler(useful for separating handlers into different files)
- `kwargs` – Optional keyword arguments(custom filters)

Результат
decorated function

`register_shipping_query_handler(`*callback: Awaitable*, *func: Callable*, *pass_ bot: Optional[bool] = False*, ***kwargs*`)`

Registers shipping query handler.

Параметры

- `callback` (`Awaitable`) – function to be called
- `func` (`function`) – Function executed as a filter
- `pass_bot` (`bool`) – True if you need to pass TeleBot instance to handler(useful for separating handlers into different files)
- `kwargs` – Optional keyword arguments(custom filters)

Результат
None

`async remove_webhook()` → bool

Alternative for delete_webhook but uses set_webhook

`async reopen_forum_topic(`*chat_ id: Union[str, int]*, *message_ thread_ id: int*`)` → bool

Use this method to reopen a closed topic in a forum supergroup chat. The bot must be an administrator in the chat for this to work and must have the can_manage_topics administrator rights, unless it is the creator of the topic. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#reopenforumtopic

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)

- `message_thread_id` (`int`) – Identifier of the topic to reopen

Результат
On success, True is returned.

Тип результата
`bool`

`async reply_to(`*message:* Message, *text: str*, ***kwargs*`)` → *Message*

Convenience function for *send_message(message.chat.id, text, reply_to_message_id=message.message_id, **kwargs)*

Параметры

- `message` (`types.Message`) – Instance of *telebot.types.Message*
- `text` (`str`) – Text of the message.
- `kwargs` – Additional keyword arguments which are passed to *telebot.TeleBot.send_message()*

Результат
On success, the sent Message is returned.

Тип результата
telebot.types.Message

`async reset_data(`*user_id: int*, *chat_id: Optional[int] = None*`)`

Reset data for a user in chat.

Параметры

- `user_id` (`int`) – User's identifier
- `chat_id` (`int`) – Chat's identifier

Результат
None

`async restrict_chat_member(`*chat_id: Union[int, str]*, *user_id: int*, *until_date: Optional[Union[int, datetime]] = None*, *can_send_messages: Optional[bool] = None*, *can_send_media_messages: Optional[bool] = None*, *can_send_polls: Optional[bool] = None*, *can_send_other_messages: Optional[bool] = None*, *can_add_web_page_previews: Optional[bool] = None*, *can_change_info: Optional[bool] = None*, *can_invite_users: Optional[bool] = None*, *can_pin_messages: Optional[bool] = None*`)` → bool

Use this method to restrict a user in a supergroup. The bot must be an administrator in the supergroup for this to work and must have the appropriate admin rights. Pass True for all boolean parameters to lift restrictions from a user.

Telegram documentation: https://core.telegram.org/bots/api#restrictchatmember

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target group or username of the target supergroup or channel (in the format @channelusername)
- `user_id` (`int`) – Unique identifier of the target user

- `until_date` (`int` or `datetime`) – Date when restrictions will be lifted for the user, unix time. If user is restricted for more than 366 days or less than 30 seconds from the current time, they are considered to be restricted forever
- `can_send_messages` (`bool`) – Pass True, if the user can send text messages, contacts, locations and venues
- `can_send_media_messages` (`bool`) – Pass True, if the user can send audios, documents, photos, videos, video notes and voice notes, implies can_send_messages
- `can_send_polls` (`bool`) – Pass True, if the user is allowed to send polls, implies can_send_messages
- `can_send_other_messages` (`bool`) – Pass True, if the user can send animations, games, stickers and use inline bots, implies can_send_media_messages
- `can_add_web_page_previews` (`bool`) – Pass True, if the user may add web page previews to their messages, implies can_send_media_messages
- `can_change_info` (`bool`) – Pass True, if the user is allowed to change the chat title, photo and other settings. Ignored in public supergroups
- `can_invite_users` (`bool`) – Pass True, if the user is allowed to invite new users to the chat, implies can_invite_users
- `can_pin_messages` (`bool`) – Pass True, if the user is allowed to pin messages. Ignored in public supergroups

Результат
True on success

Тип результата
`bool`

`retrieve_data`(*user_id: int*, *chat_id: Optional[int] = None*)

Returns context manager with data for a user in chat.

Параметры

- `user_id` (*int*) – User identifier
- `chat_id` (*int, optional*) – Chat's unique identifier, defaults to user_id

Результат
Context manager with data for a user in chat

Тип результата
Optional[Any]

`async revoke_chat_invite_link`(*chat_id: Union[int, str]*, *invite_link: str*) → *ChatInviteLink*

Use this method to revoke an invite link created by the bot. Note: If the primary link is revoked, a new link is automatically generated The bot must be an administrator in the chat for this to work and must have the appropriate admin rights.

Telegram documentation: https://core.telegram.org/bots/api#revokechatinvitelink

Параметры

- `chat_id` (`int` or `str`) – Id: Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `invite_link` (`str`) – The invite link to revoke

Результат
Returns the new invite link as ChatInviteLink object.

Тип результата
telebot.types.ChatInviteLink

async run_webhooks(*listen: Optional[str] = '127.0.0.1', port: Optional[int] = 443, url_path: Optional[str] = None, certificate: Optional[str] = None, certificate_key: Optional[str] = None, webhook_url: Optional[str] = None, max_connections: Optional[int] = None, allowed_updates: Optional[List] = None, ip_address: Optional[str] = None, drop_pending_updates: Optional[bool] = None, timeout: Optional[int] = None, secret_token: Optional[str] = None, secret_token_length: Optional[int] = 20, debug: Optional[bool] = False*)

This class sets webhooks and listens to a given url and port.

Параметры

- `listen` – IP address to listen to. Defaults to 0.0.0.0
- `port` – A port which will be used to listen to webhooks.
- `url_path` – Path to the webhook. Defaults to /token
- `certificate` – Path to the certificate file.
- `certificate_key` – Path to the certificate key file.
- `webhook_url` – Webhook URL.
- `max_connections` – Maximum allowed number of simultaneous HTTPS connections to the webhook for update delivery, 1-100. Defaults to 40. Use lower values to limit the load on your bot's server, and higher values to increase your bot's throughput.
- `allowed_updates` – A JSON-serialized list of the update types you want your bot to receive. For example, specify [“message”, “edited_channel_post”, “callback_query”] to only receive updates of these types. See Update for a complete list of available update types. Specify an empty list to receive all updates regardless of type (default). If not specified, the previous setting will be used.
- `ip_address` – The fixed IP address which will be used to send webhook requests instead of the IP address resolved through DNS
- `drop_pending_updates` – Pass True to drop all pending updates
- `timeout` – Integer. Request connection timeout
- `secret_token` – Secret token to be used to verify the webhook request.

Результат

async send_animation(*chat_id: Union[int, str], animation: Union[Any, str], duration: Optional[int] = None, width: Optional[int] = None, height: Optional[int] = None, thumb: Optional[Union[str, Any]] = None, caption: Optional[str] = None, parse_mode: Optional[str] = None, caption_entities: Optional[List[MessageEntity]] = None, disable_notification: Optional[bool] = None, protect_content: Optional[bool] = None, reply_to_message_id: Optional[int] = None, allow_sending_without_reply: Optional[bool] = None, reply_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None, timeout: Optional[int] = None, message_thread_id: Optional[int] = None*) → *Message*

Use this method to send animation files (GIF or H.264/MPEG-4 AVC video without sound). On success, the sent Message is returned. Bots can currently send animation files of up to 50 MB in size, this limit may be changed in the future.

Telegram documentation: https://core.telegram.org/bots/api#sendanimation

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `animation` (`str` or *telebot.types.InputFile*) – Animation to send. Pass a file_id as String to send an animation that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get an animation from the Internet, or upload a new animation using multipart/form-data.
- `duration` (`int`) – Duration of sent animation in seconds
- `width` (`int`) – Animation width
- `height` (`int`) – Animation height
- `thumb` (`str` or *telebot.types.InputFile*) – Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass "attach://<file_attach_name>" if the thumbnail was uploaded using multipart/form-data under <file_attach_name>.
- `caption` (`str`) – Animation caption (may also be used when resending animation by file_id), 0-1024 characters after entities parsing
- `parse_mode` (`str`) – Mode for parsing entities in the animation caption
- `protect_content` (`bool`) – Protects the contents of the sent message from forwarding and saving
- `reply_to_message_id` (`int`) – If the message is a reply, ID of the original message
- `reply_markup` (*telebot.types.InlineKeyboardMarkup* or *telebot.types.ReplyKeyboardMarkup* or *telebot.types.ReplyKeyboardRemove* or *telebot.types.ForceReply*) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user.
- `disable_notification` (`bool`) – Sends the message silently. Users will receive a notification with no sound.
- `timeout` (`int`) – Timeout in seconds for the request.
- `caption_entities` (`list` of *telebot.types.MessageEntity*) – List of special entities that appear in the caption, which can be specified instead of parse_mode
- `allow_sending_without_reply` (`bool`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- `message_thread_id` (`int`) – Identifier of a message thread, in which the video will be sent

Результат

On success, the sent Message is returned.

Тип результата
telebot.types.Message

async send_audio(*chat_id: Union[int, str]*, *audio: Union[Any, str]*, *caption: Optional[str] = None*, *duration: Optional[int] = None*, *performer: Optional[str] = None*, *title: Optional[str] = None*, *reply_to_message_id: Optional[int] = None*, *reply_markup: Optional[Union[*InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply*]] = None*, *parse_mode: Optional[str] = None*, *disable_notification: Optional[bool] = None*, *timeout: Optional[int] = None*, *thumb: Optional[Union[str, Any]] = None*, *caption_entities: Optional[List[*MessageEntity*]] = None*, *allow_sending_without_reply: Optional[bool] = None*, *protect_content: Optional[bool] = None*, *message_thread_id: Optional[int] = None*) → *Message*

Use this method to send audio files, if you want Telegram clients to display them in the music player. Your audio must be in the .MP3 or .M4A format. On success, the sent Message is returned. Bots can currently send audio files of up to 50 MB in size, this limit may be changed in the future.

For sending voice messages, use the send_voice method instead.

Telegram documentation: https://core.telegram.org/bots/api#sendaudio

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `audio` (`str` or *telebot.types.InputFile*) – Audio file to send. Pass a file_id as String to send an audio file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get an audio file from the Internet, or upload a new one using multipart/form-data. Audio must be in the .MP3 or .M4A format.
- `caption` (`str`) – Audio caption, 0-1024 characters after entities parsing
- `duration` (`int`) – Duration of the audio in seconds
- `performer` (`str`) – Performer
- `title` (`str`) – Track name
- `reply_to_message_id` (`int`) – If the message is a reply, ID of the original message
- `reply_markup` (*telebot.types.InlineKeyboardMarkup* or *telebot.types.ReplyKeyboardMarkup* or *telebot.types.ReplyKeyboardRemove* or *telebot.types.ForceReply*) –
- `parse_mode` (`str`) – Mode for parsing entities in the audio caption. See formatting options for more details.
- `disable_notification` (`bool`) – Sends the message silently. Users will receive a notification with no sound.
- `timeout` (`int`) – Timeout in seconds for the request.
- `thumb` (`str`) – Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass "attach://<file_attach_name>" if the thumbnail was uploaded using multipart/form-data under <file_attach_name>

- `caption_entities` (`list` of *telebot.types.MessageEntity*) – A JSON-serialized list of special entities that appear in the caption, which can be specified instead of parse_mode
- `allow_sending_without_reply` (`bool`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- `protect_content` (`bool`) – Protects the contents of the sent message from forwarding and saving
- `message_thread_id` (`int`) – Identifier of a message thread, in which the message will be sent

Результат
On success, the sent Message is returned.

Тип результата
telebot.types.Message

async send_chat_action(*chat_id: Union[int, str]*, *action: str*, *timeout: Optional[int] = None*) → bool

Use this method when you need to tell the user that something is happening on the bot's side. The status is set for 5 seconds or less (when a message arrives from your bot, Telegram clients clear its typing status). Returns True on success.

Example: The ImageBot needs some time to process a request and upload the image. Instead of sending a text message along the lines of "Retrieving image, please wait...", the bot may use sendChatAction with action = upload_photo. The user will see a "sending photo" status for the bot.

Telegram documentation: https://core.telegram.org/bots/api#sendchataction

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel
- `action` (`str`) – Type of action to broadcast. Choose one, depending on what the user is about to receive: typing for text messages, upload_photo for photos, record_video or upload_video for videos, record_voice or upload_voice for voice notes, upload_document for general files, choose_sticker for stickers, find_location for location data, record_video_note or upload_video_note for video notes.
- `timeout` (`int`) – Timeout in seconds for the request.

Результат
Returns True on success.

Тип результата
`bool`

async send_contact(*chat_id: Union[int, str]*, *phone_number: str*, *first_name: str*, *last_name: Optional[str] = None*, *vcard: Optional[str] = None*, *disable_notification: Optional[bool] = None*, *reply_to_message_id: Optional[int] = None*, *reply_markup: Optional[Union[*InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply*]] = None*, *timeout: Optional[int] = None*, *allow_sending_without_reply: Optional[bool] = None*, *protect_content: Optional[bool] = None*, *message_thread_id: Optional[int] = None*) → *Message*

Use this method to send phone contacts. On success, the sent Message is returned.

Telegram documentation: https://core.telegram.org/bots/api#sendcontact

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel
- `phone_number` (`str`) – Contact's phone number
- `first_name` (`str`) – Contact's first name
- `last_name` (`str`) – Contact's last name
- `vcard` (`str`) – Additional data about the contact in the form of a vCard, 0-2048 bytes
- `disable_notification` (`bool`) – Sends the message silently. Users will receive a notification with no sound.
- `reply_to_message_id` (`int`) – If the message is a reply, ID of the original message
- `reply_markup` (*telebot.types.InlineKeyboardMarkup* or *telebot.types.ReplyKeyboardMarkup* or *telebot.types.ReplyKeyboardRemove* or *telebot.types.ForceReply*) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user.
- `timeout` (`int`) – Timeout in seconds for the request.
- `allow_sending_without_reply` (`bool`) – Pass True, if the message should be sent even if one of the specified replied-to messages is not found.
- `protect_content` (`bool`) – Protects the contents of the sent message from forwarding and saving
- `message_thread_id` (`int`) – The thread to which the message will be sent

Результат

On success, the sent Message is returned.

Тип результата

telebot.types.Message

async send_dice(*chat_id: Union[int, str]*, *emoji: Optional[str] = None*, *disable_notification: Optional[bool] = None*, *reply_to_message_id: Optional[int] = None*, *reply_markup: Optional[Union[*InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply*]] = None*, *timeout: Optional[int] = None*, *allow_sending_without_reply: Optional[bool] = None*, *protect_content: Optional[bool] = None*, *message_thread_id: Optional[int] = None*) → *Message*

Use this method to send an animated emoji that will display a random value. On success, the sent Message is returned.

Telegram documentation: https://core.telegram.org/bots/api#senddice

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `emoji` (`str`) – Emoji on which the dice throw animation is based. Currently, must be one of “”, “”, “”, “”, “”, or “”. Dice can have values 1-6 for “”, “” and “”, values 1-5 for “” and “”, and values 1-64 for “”. Defaults to “”

- `disable_notification` (`bool`) – Sends the message silently. Users will receive a notification with no sound.
- `reply_to_message_id` (`int`) – If the message is a reply, ID of the original message
- `reply_markup` (*telebot.types.InlineKeyboardMarkup* or *telebot.types.ReplyKeyboardMarkup* or *telebot.types.ReplyKeyboardRemove* or *telebot.types.ForceReply*) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user.
- `timeout` (`int`) – Timeout in seconds for the request.
- `allow_sending_without_reply` (`bool`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- `protect_content` (`bool`) – Protects the contents of the sent message from forwarding
- `message_thread_id` (`int`) – The identifier of a message thread, unique within the chat to which the message with the thread identifier belongs

Результат
On success, the sent Message is returned.

Тип результата
telebot.types.Message

async send_document(*chat_id: Union[int, str]*, *document: Union[Any, str]*, *reply_to_message_id: Optional[int] = None*, *caption: Optional[str] = None*, *reply_markup: Optional[Union[*InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply*]] = None*, *parse_mode: Optional[str] = None*, *disable_notification: Optional[bool] = None*, *timeout: Optional[int] = None*, *thumb: Optional[Union[str, Any]] = None*, *caption_entities: Optional[List[*MessageEntity*]] = None*, *allow_sending_without_reply: Optional[bool] = None*, *visible_file_name: Optional[str] = None*, *disable_content_type_detection: Optional[bool] = None*, *data: Optional[Union[str, Any]] = None*, *protect_content: Optional[bool] = None*, *message_thread_id: Optional[int] = None*) → *Message*

Use this method to send general files.

Telegram documentation: https://core.telegram.org/bots/api#senddocument

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `document` (`str` or *telebot.types.InputFile*) – (document) File to send. Pass a file_id as String to send a file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a file from the Internet, or upload a new one using multipart/form-data
- `reply_to_message_id` (`int`) – If the message is a reply, ID of the original message
- `caption` (`str`) – Document caption (may also be used when resending documents by file_id), 0-1024 characters after entities parsing
- `reply_markup` (*telebot.types.InlineKeyboardMarkup* or *telebot.types.ReplyKeyboardMarkup* or *telebot.types.ReplyKeyboardRemove* or *telebot.types.ForceReply*) – Additional interface options. A JSON-serialized object for

an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user.

- `parse_mode` (`str`) – Mode for parsing entities in the document caption
- `disable_notification` (`bool`) – Sends the message silently. Users will receive a notification with no sound.
- `timeout` (`int`) – Timeout in seconds for the request.
- `thumb` (`str` or *telebot.types.InputFile*) – InputFile or String : Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass "attach://<file_attach_name>" if the thumbnail was uploaded using multipart/form-data under <file_attach_name>
- `caption_entities` (`list` of *telebot.types.MessageEntity*) – A JSON-serialized list of special entities that appear in the caption, which can be specified instead of parse_mode
- `allow_sending_without_reply` (`bool`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- `visible_file_name` (`str`) – allows to define file name that will be visible in the Telegram instead of original file name
- `disable_content_type_detection` (`bool`) – Disables automatic server-side content type detection for files uploaded using multipart/form-data
- `data` (`str`) – function typo miss compatibility: do not use it
- `protect_content` (`bool`) – Protects the contents of the sent message from forwarding and saving
- `message_thread_id` (`int`) – Identifier of a message thread, in which the message will be sent

Результат

On success, the sent Message is returned.

Тип результата

telebot.types.Message

async send_game(*chat_id: Union[int, str]*, *game_short_name: str*, *disable_notification: Optional[bool] = None*, *reply_to_message_id: Optional[int] = None*, *reply_markup: Optional[Union[*InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply*]] = None*, *timeout: Optional[int] = None*, *allow_sending_without_reply: Optional[bool] = None*, *protect_content: Optional[bool] = None*, *message_thread_id: Optional[int] = None*) → *Message*

Used to send the game.

Telegram documentation: https://core.telegram.org/bots/api#sendgame

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `game_short_name` (`str`) – Short name of the game, serves as the unique identifier for the game. Set up your games via @BotFather.

- `disable_notification` (`bool`) – Sends the message silently. Users will receive a notification with no sound.
- `reply_to_message_id` (`int`) – If the message is a reply, ID of the original message
- `reply_markup` (`InlineKeyboardMarkup` or `ReplyKeyboardMarkup` or `ReplyKeyboardRemove` or `ForceReply`) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user.
- `timeout` (`int`) – Timeout in seconds for waiting for a response from the bot.
- `allow_sending_without_reply` (`bool`) – Pass True, if the message should be sent even if one of the specified replied-to messages is not found.
- `protect_content` (`bool`) – Pass True, if content of the message needs to be protected from being viewed by the bot.
- `message_thread_id` (`int`) – Identifier of the thread to which the message will be sent.

Результат
On success, the sent Message is returned.

Тип результата
`types.Message`

async send_invoice(*chat_id: Union[int, str]*, *title: str*, *description: str*, *invoice_payload: str*, *provider_token: str*, *currency: str*, *prices: List[LabeledPrice]*, *start_parameter: Optional[str] = None*, *photo_url: Optional[str] = None*, *photo_size: Optional[int] = None*, *photo_width: Optional[int] = None*, *photo_height: Optional[int] = None*, *need_name: Optional[bool] = None*, *need_phone_number: Optional[bool] = None*, *need_email: Optional[bool] = None*, *need_shipping_address: Optional[bool] = None*, *send_phone_number_to_provider: Optional[bool] = None*, *send_email_to_provider: Optional[bool] = None*, *is_flexible: Optional[bool] = None*, *disable_notification: Optional[bool] = None*, *reply_to_message_id: Optional[int] = None*, *reply_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None*, *provider_data: Optional[str] = None*, *timeout: Optional[int] = None*, *allow_sending_without_reply: Optional[bool] = None*, *max_tip_amount: Optional[int] = None*, *suggested_tip_amounts: Optional[List[int]] = None*, *protect_content: Optional[bool] = None*, *message_thread_id: Optional[int] = None*) → *Message*

Sends invoice.

Telegram documentation: https://core.telegram.org/bots/api#sendinvoice

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target private chat
- `title` (`str`) – Product name, 1-32 characters
- `description` (`str`) – Product description, 1-255 characters
- `invoice_payload` (`str`) – Bot-defined invoice payload, 1-128 bytes. This will not be displayed to the user, use for your internal processes.
- `provider_token` (`str`) – Payments provider token, obtained via @Botfather

- currency (str) – Three-letter ISO 4217 currency code, see https://core.telegram.org/bots/payments#supported-currencies
- prices (List[types.LabeledPrice]) – Price breakdown, a list of components (e.g. product price, tax, discount, delivery cost, delivery tax, bonus, etc.)
- start_parameter (str) – Unique deep-linking parameter that can be used to generate this invoice when used as a start parameter
- photo_url (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.
- photo_size (int) – Photo size in bytes
- photo_width (int) – Photo width
- photo_height (int) – Photo height
- need_name (bool) – Pass True, if you require the user's full name to complete the order
- need_phone_number (bool) – Pass True, if you require the user's phone number to complete the order
- need_email (bool) – Pass True, if you require the user's email to complete the order
- need_shipping_address (bool) – Pass True, if you require the user's shipping address to complete the order
- is_flexible (bool) – Pass True, if the final price depends on the shipping method
- send_phone_number_to_provider (bool) – Pass True, if user's phone number should be sent to provider
- send_email_to_provider (bool) – Pass True, if user's email address should be sent to provider
- disable_notification (bool) – Sends the message silently. Users will receive a notification with no sound.
- reply_to_message_id (int) – If the message is a reply, ID of the original message
- reply_markup (str) – A JSON-serialized object for an inline keyboard. If empty, one „Pay total price“ button will be shown. If not empty, the first button must be a Pay button
- provider_data (str) – A JSON-serialized data about the invoice, which will be shared with the payment provider. A detailed description of required fields should be provided by the payment provider.
- timeout (int) – Timeout of a request, defaults to None
- allow_sending_without_reply (bool) – Pass True, if the message should be sent even if the specified replied-to message is not found
- max_tip_amount (int) – The maximum accepted amount for tips in the smallest units of the currency
- suggested_tip_amounts (list of int) – A JSON-serialized array of suggested amounts of tips in the smallest units of the currency. At most 4 suggested tip amounts can be specified. The suggested tip amounts must be positive, passed in a strictly increased order and must not exceed max_tip_amount.

- `protect_content` (`bool`) – Protects the contents of the sent message from forwarding and saving
- `message_thread_id` (`int`) – The identifier of a message thread, in which the invoice message will be sent

Результат
On success, the sent Message is returned.

Тип результата
`types.Message`

async send_location(*chat_id: Union[int, str]*, *latitude: float*, *longitude: float*, *live_period: Optional[int] = None*, *reply_to_message_id: Optional[int] = None*, *reply_markup: Optional[Union[*InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply*]] = None*, *disable_notification: Optional[bool] = None*, *timeout: Optional[int] = None*, *horizontal_accuracy: Optional[float] = None*, *heading: Optional[int] = None*, *proximity_alert_radius: Optional[int] = None*, *allow_sending_without_reply: Optional[bool] = None*, *protect_content: Optional[bool] = None*, *message_thread_id: Optional[int] = None*) → *Message*

Use this method to send point on the map. On success, the sent Message is returned.

Telegram documentation: https://core.telegram.org/bots/api#sendlocation

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `latitude` (`float`) – Latitude of the location
- `longitude` (`float`) – Longitude of the location
- `live_period` (`int`) – Period in seconds for which the location will be updated (see Live Locations, should be between 60 and 86400.
- `reply_to_message_id` (`int`) – If the message is a reply, ID of the original message
- `reply_markup` (*telebot.types.InlineKeyboardMarkup* or *telebot.types.ReplyKeyboardMarkup* or *telebot.types.ReplyKeyboardRemove* or *telebot.types.ForceReply*) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user.
- `disable_notification` (`bool`) – Sends the message silently. Users will receive a notification with no sound.
- `timeout` (`int`) – Timeout in seconds for the request.
- `horizontal_accuracy` (`float`) – The radius of uncertainty for the location, measured in meters; 0-1500
- `heading` (`int`) – For live locations, a direction in which the user is moving, in degrees. Must be between 1 and 360 if specified.
- `proximity_alert_radius` (`int`) – For live locations, a maximum distance for proximity alerts about approaching another chat member, in meters. Must be between 1 and 100000 if specified.
- `allow_sending_without_reply` (`bool`) – Pass True, if the message should be sent even if the specified replied-to message is not found

- `protect_content` (`bool`) – Protects the contents of the sent message from forwarding and saving
- `message_thread_id` (`int`) – Identifier of a message thread, in which the message will be sent

Результат
On success, the sent Message is returned.

Тип результата
telebot.types.Message

async send_media_group(*chat_id: Union[int, str]*, *media: List[Union[*InputMediaAudio, InputMediaDocument, InputMediaPhoto, InputMediaVideo*]]*, *disable_notification: Optional[bool] = None*, *protect_content: Optional[bool] = None*, *reply_to_message_id: Optional[int] = None*, *timeout: Optional[int] = None*, *allow_sending_without_reply: Optional[bool] = None*, *message_thread_id: Optional[int] = None*) → List[*Message*]

Use this method to send a group of photos, videos, documents or audios as an album. Documents and audio files can be only grouped in an album with messages of the same type. On success, an array of Messages that were sent is returned.

Telegram documentation: https://core.telegram.org/bots/api#sendmediagroup

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `media` (`list` of `types.InputMedia`) – A JSON-serialized array describing messages to be sent, must include 2-10 items
- `disable_notification` (`bool`) – Sends the messages silently. Users will receive a notification with no sound.
- `protect_content` (`bool`) – Protects the contents of the sent message from forwarding and saving
- `reply_to_message_id` (`int`) – If the message is a reply, ID of the original message
- `timeout` (`int`) – Timeout in seconds for the request.
- `allow_sending_without_reply` (`bool`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- `message_thread_id` (`int`) – Identifier of a message thread, in which the messages will be sent

Результат
On success, an array of Messages that were sent is returned.

Тип результата
List[*types.Message*]

async send_message(*chat_id: Union[int, str]*, *text: str*, *parse_mode: Optional[str] = None*, *entities: Optional[List[*MessageEntity*]] = None*, *disable_web_page_preview: Optional[bool] = None*, *disable_notification: Optional[bool] = None*, *protect_content: Optional[bool] = None*, *reply_to_message_id: Optional[int] = None*, *allow_sending_without_reply: Optional[bool] = None*, *reply_markup: Optional[Union[*InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply*]] = None*, *timeout: Optional[int] = None*, *message_thread_id: Optional[int] = None*) → *Message*

Use this method to send text messages.

Warning: Do not send more than about 4096 characters each message, otherwise you'll risk an HTTP 414 error. If you must send more than 4096 characters, use the *split_string* or *smart_split* function in util.py.

Telegram documentation: https://core.telegram.org/bots/api#sendmessage

Параметры

- `chat_id` (`int` or `str`) – 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
- `parse_mode` (`str`) – Mode for parsing entities in the message text.
- `entities` (Array of *telebot.types.MessageEntity*) – List of special entities that appear in message text, which can be specified instead of parse_mode
- `disable_web_page_preview` (`bool`) – Disables link previews for links in this message
- `disable_notification` (`bool`) – Sends the message silently. Users will receive a notification with no sound.
- `protect_content` (`bool`) – If True, the message content will be hidden for all users except for the target user
- `reply_to_message_id` (`int`) – If the message is a reply, ID of the original message
- `allow_sending_without_reply` (`bool`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- `reply_markup` (*telebot.types.InlineKeyboardMarkup* or *telebot.types.ReplyKeyboardMarkup* or *telebot.types.ReplyKeyboardRemove* or *telebot.types.ForceReply*) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user.
- `timeout` (`int`) – Timeout in seconds for the request.
- `message_thread_id` (`int`) – Unique identifier for the target message thread (topic) of the forum; for forum supergroups only

Результат
On success, the sent Message is returned.

Тип результата
telebot.types.Message

`async send_photo(`*chat_id: Union[int, str]*, *photo: Union[Any, str]*, *caption: Optional[str] = None*, *parse_mode: Optional[str] = None*, *caption_entities: Optional[List[MessageEntity]] = None*, *disable_notification: Optional[bool] = None*, *protect_content: Optional[bool] = None*, *reply_to_message_id: Optional[int] = None*, *allow_sending_without_reply: Optional[bool] = None*, *reply_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None*, *timeout: Optional[int] = None*, *message_thread_id: Optional[int] = None*`)` → *Message*

Use this method to send photos. On success, the sent Message is returned.

Telegram documentation: https://core.telegram.org/bots/api#sendphoto

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `photo` (`str` or *telebot.types.InputFile*) – Photo to send. Pass a file_id as String to send a photo that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a photo from the Internet, or upload a new photo using multipart/form-data. The photo must be at most 10 MB in size. The photo's width and height must not exceed 10000 in total. Width and height ratio must be at most 20.
- `caption` (`str`) – Photo caption (may also be used when resending photos by file_id), 0-1024 characters after entities parsing
- `parse_mode` (`str`) – Mode for parsing entities in the photo caption.
- `caption_entities` (`list` of *telebot.types.MessageEntity*) – A JSON-serialized list of special entities that appear in the caption, which can be specified instead of parse_mode
- `disable_notification` (`bool`) – Sends the message silently. Users will receive a notification with no sound.
- `protect_content` (`bool`) – Protects the contents of the sent message from forwarding and saving
- `reply_to_message_id` (`int`) – If the message is a reply, ID of the original message
- `allow_sending_without_reply` (`bool`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- `reply_markup` (*telebot.types.InlineKeyboardMarkup* or *telebot.types.ReplyKeyboardMarkup* or *telebot.types.ReplyKeyboardRemove* or *telebot.types.ForceReply*) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user.
- `timeout` (`int`) – Timeout in seconds for the request.
- `message_thread_id` (`int`) – Identifier of a message thread, in which the message will be sent

Результат

On success, the sent Message is returned.

Тип результата

telebot.types.Message

async send_poll(*chat_id: Union[int, str]*, *question: str*, *options: List[str]*, *is_anonymous: Optional[bool] = None*, *type: Optional[str] = None*, *allows_multiple_answers: Optional[bool] = None*, *correct_option_id: Optional[int] = None*, *explanation: Optional[str] = None*, *explanation_parse_mode: Optional[str] = None*, *open_period: Optional[int] = None*, *close_date: Optional[Union[int, datetime]] = None*, *is_closed: Optional[bool] = None*, *disable_notification: Optional[bool] = False*, *reply_to_message_id: Optional[int] = None*, *reply_markup: Optional[Union[*InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply*]] = None*, *allow_sending_without_reply: Optional[bool] = None*, *timeout: Optional[int] = None*, *explanation_entities: Optional[List[*MessageEntity*]] = None*, *protect_content: Optional[bool] = None*, *message_thread_id: Optional[int] = None*) → *Message*

Use this method to send a native poll. On success, the sent Message is returned.

Telegram documentation: https://core.telegram.org/bots/api#sendpoll

Параметры

- `chat_id` (`int` | `str`) – Unique identifier for the target chat or username of the target channel
- `question` (`str`) – Poll question, 1-300 characters
- `options` (`list` of `str`) – A JSON-serialized list of answer options, 2-10 strings 1-100 characters each
- `is_anonymous` (`bool`) – True, if the poll needs to be anonymous, defaults to True
- `type` (`str`) – Poll type, "quiz" or "regular", defaults to "regular"
- `allows_multiple_answers` (`bool`) – True, if the poll allows multiple answers, ignored for polls in quiz mode, defaults to False
- `correct_option_id` (`int`) – 0-based identifier of the correct answer option. Available only for polls in quiz mode, defaults to None
- `explanation` (`str`) – Text that is shown when a user chooses an incorrect answer or taps on the lamp icon in a quiz-style poll, 0-200 characters with at most 2 line feeds after entities parsing
- `explanation_parse_mode` (`str`) – Mode for parsing entities in the explanation. See formatting options for more details.
- `open_period` (`int`) – Amount of time in seconds the poll will be active after creation, 5-600. Can't be used together with close_date.
- `close_date` (`int` | `datetime`) – Point in time (Unix timestamp) when the poll will be automatically closed.
- `is_closed` (`bool`) – Pass True, if the poll needs to be immediately closed. This can be useful for poll preview.
- `disable_notification` (`bool`) – Sends the message silently. Users will receive a notification with no sound.
- `reply_to_message_id` (`int`) – If the message is a reply, ID of the original message
- `allow_sending_without_reply` (`bool`) – Pass True, if the poll allows multiple options to be voted simultaneously.
- `reply_markup` (`InlineKeyboardMarkup` | `ReplyKeyboardMarkup` | `ReplyKeyboardRemove` | `ForceReply`) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user.
- `timeout` (`int`) – Timeout in seconds for waiting for a response from the user.
- `explanation_entities` (`list` of `MessageEntity`) – A JSON-serialized list of special entities that appear in the explanation, which can be specified instead of parse_mode
- `protect_content` (`bool`) – Protects the contents of the sent message from forwarding and saving
- `message_thread_id` (`int`) – The identifier of a message thread, in which the poll will be sent

Результат
On success, the sent Message is returned.

Тип результата
types.Message

async send_sticker(*chat_id: Union[int, str]*, *sticker: Union[Any, str]*, *reply_to_message_id: Optional[int] = None*, *reply_markup: Optional[Union[*InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply*]] = None*, *disable_notification: Optional[bool] = None*, *timeout: Optional[int] = None*, *allow_sending_without_reply: Optional[bool] = None*, *protect_content: Optional[bool] = None*, *data: Optional[Union[str, Any]] = None*, *message_thread_id: Optional[int] = None*) → *Message*

Use this method to send static .WEBP, animated .TGS, or video .WEBM stickers. On success, the sent Message is returned.

Telegram documentation: https://core.telegram.org/bots/api#sendsticker

Параметры

- chat_id (int or str) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- sticker (str or *telebot.types.InputFile*) – Sticker to send. Pass a file_id as String to send a file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a .webp file from the Internet, or upload a new one using multipart/form-data.
- reply_to_message_id (int) – If the message is a reply, ID of the original message
- reply_markup (*telebot.types.InlineKeyboardMarkup* or *telebot.types.ReplyKeyboardMarkup* or *telebot.types.ReplyKeyboardRemove* or *telebot.types.ForceReply*) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user.
- disable_notification (bool) – to disable the notification
- timeout (int) – Timeout in seconds for the request.
- allow_sending_without_reply (bool) – Pass True, if the message should be sent even if the specified replied-to message is not found
- protect_content (bool) – Protects the contents of the sent message from forwarding and saving
- data (str) – function typo miss compatibility: do not use it
- message_thread_id (int) – Identifier of a message thread, in which the message will be sent

Результат
On success, the sent Message is returned.

Тип результата
telebot.types.Message

async send_venue(*chat_id: Union[int, str]*, *latitude: float*, *longitude: float*, *title: str*, *address: str*, *foursquare_id: Optional[str] = None*, *foursquare_type: Optional[str] = None*, *disable_notification: Optional[bool] = None*, *reply_to_message_id: Optional[int] = None*, *reply_markup: Optional[Union[*InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply*]] = None*, *timeout: Optional[int] = None*, *allow_sending_without_reply: Optional[bool] = None*, *google_place_id: Optional[str] = None*, *google_place_type: Optional[str] = None*, *protect_content: Optional[bool] = None*, *message_thread_id: Optional[int] = None*) → *Message*

Use this method to send information about a venue. On success, the sent Message is returned.

Telegram documentation: https://core.telegram.org/bots/api#sendvenue

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel
- `latitude` (`float`) – Latitude of the venue
- `longitude` (`float`) – Longitude of the venue
- `title` (`str`) – Name of the venue
- `address` (`str`) – Address of the venue
- `foursquare_id` (`str`) – Foursquare identifier of the venue
- `foursquare_type` (`str`) – Foursquare type of the venue, if known. (For example, “arts_entertainment/default”, “arts_entertainment/aquarium” or “food/icecream”.)
- `disable_notification` (`bool`) – Sends the message silently. Users will receive a notification with no sound.
- `reply_to_message_id` (`int`) – If the message is a reply, ID of the original message
- `reply_markup` (*`telebot.types.InlineKeyboardMarkup`* or *`telebot.types.ReplyKeyboardMarkup`* or *`telebot.types.ReplyKeyboardRemove`* or *`telebot.types.ForceReply`*) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user.
- `timeout` (`int`) – Timeout in seconds for the request.
- `allow_sending_without_reply` (`bool`) – Pass True, if the message should be sent even if one of the specified replied-to messages is not found.
- `google_place_id` (`str`) – Google Places identifier of the venue
- `google_place_type` (`str`) – Google Places type of the venue.
- `protect_content` (`bool`) – Protects the contents of the sent message from forwarding and saving
- `message_thread_id` (`int`) – The thread to which the message will be sent

Результат

On success, the sent Message is returned.

Тип результата

`telebot.types.Message`

async send_video(*chat_id: Union[int, str]*, *video: Union[Any, str]*, *duration: Optional[int] = None*, *width: Optional[int] = None*, *height: Optional[int] = None*, *thumb: Optional[Union[str, Any]] = None*, *caption: Optional[str] = None*, *parse_mode: Optional[str] = None*, *caption_entities: Optional[List[MessageEntity]] = None*, *supports_streaming: Optional[bool] = None*, *disable_notification: Optional[bool] = None*, *protect_content: Optional[bool] = None*, *reply_to_message_id: Optional[int] = None*, *allow_sending_without_reply: Optional[bool] = None*, *reply_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None*, *timeout: Optional[int] = None*, *data: Optional[Union[str, Any]] = None*, *message_thread_id: Optional[int] = None*) → *Message*

Use this method to send video files, Telegram clients support mp4 videos (other formats may be sent as Document).

Telegram documentation: https://core.telegram.org/bots/api#sendvideo

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `video` (`str` or *`telebot.types.InputFile`*) – Video to send. You can either pass a file_id as String to resend a video that is already on the Telegram servers, or upload a new video file using multipart/form-data.
- `duration` (`int`) – Duration of sent video in seconds
- `width` (`int`) – Video width
- `height` (`int`) – Video height
- `thumb` (`str` or *`telebot.types.InputFile`*) – Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass "attach://<file_attach_name>" if the thumbnail was uploaded using multipart/form-data under <file_attach_name>.
- `caption` (`str`) – Video caption (may also be used when resending videos by file_id), 0-1024 characters after entities parsing
- `parse_mode` (`str`) – Mode for parsing entities in the video caption
- `caption_entities` (`list` of *`telebot.types.MessageEntity`*) – List of special entities that appear in the caption, which can be specified instead of parse_mode
- `supports_streaming` (`bool`) – Pass True, if the uploaded video is suitable for streaming
- `disable_notification` (`bool`) – Sends the message silently. Users will receive a notification with no sound.
- `protect_content` (`bool`) – Protects the contents of the sent message from forwarding and saving
- `reply_to_message_id` (`int`) – If the message is a reply, ID of the original message
- `allow_sending_without_reply` (`bool`) – Pass True, if the message should be sent even if the specified replied-to message is not found

- reply_markup (*telebot.types.InlineKeyboardMarkup* or *telebot.types.ReplyKeyboardMarkup* or *telebot.types.ReplyKeyboardRemove* or *telebot.types.ForceReply*) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user.
- timeout (int) – Timeout in seconds for the request.
- data (str) – function typo miss compatibility: do not use it
- message_thread_id (int) – Identifier of a message thread, in which the video will be sent

Результат
On success, the sent Message is returned.

Тип результата
telebot.types.Message

async send_video_note(*chat_id: Union[int, str]*, *data: Union[Any, str]*, *duration: Optional[int] = None*, *length: Optional[int] = None*, *reply_to_message_id: Optional[int] = None*, *reply_markup: Optional[Union[*InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply*]] = None*, *disable_notification: Optional[bool] = None*, *timeout: Optional[int] = None*, *thumb: Optional[Union[str, Any]] = None*, *allow_sending_without_reply: Optional[bool] = None*, *protect_content: Optional[bool] = None*, *message_thread_id: Optional[int] = None*) → *Message*

As of v.4.0, Telegram clients support rounded square MPEG4 videos of up to 1 minute long. Use this method to send video messages. On success, the sent Message is returned.

Telegram documentation: https://core.telegram.org/bots/api#sendvideonote

Параметры

- chat_id (int or str) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- data (str or *telebot.types.InputFile*) – Video note to send. Pass a file_id as String to send a video note that exists on the Telegram servers (recommended) or upload a new video using multipart/form-data. Sending video notes by a URL is currently unsupported
- duration (int) – Duration of sent video in seconds
- length (int) – Video width and height, i.e. diameter of the video message
- reply_to_message_id (int) – If the message is a reply, ID of the original message
- reply_markup (*telebot.types.InlineKeyboardMarkup* or *telebot.types.ReplyKeyboardMarkup* or *telebot.types.ReplyKeyboardRemove* or *telebot.types.ForceReply*) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user.
- disable_notification (bool) – Sends the message silently. Users will receive a notification with no sound.
- timeout (int) – Timeout in seconds for the request.
- thumb (str or *telebot.types.InputFile*) – Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width

and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass "attach://<file_attach_name>" if the thumbnail was uploaded using multipart/form-data under <file_attach_name>.

- `allow_sending_without_reply` (`bool`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- `protect_content` (`bool`) – Protects the contents of the sent message from forwarding and saving
- `message_thread_id` (`int`) – Identifier of a message thread, in which the video note will be sent

Результат
On success, the sent Message is returned.

Тип результата
telebot.types.Message

async send_voice(*chat_id: Union[int, str]*, *voice: Union[Any, str]*, *caption: Optional[str] = None*, *duration: Optional[int] = None*, *reply_to_message_id: Optional[int] = None*, *reply_markup: Optional[Union[*InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply*]] = None*, *parse_mode: Optional[str] = None*, *disable_notification: Optional[bool] = None*, *timeout: Optional[int] = None*, *caption_entities: Optional[List[*MessageEntity*]] = None*, *allow_sending_without_reply: Optional[bool] = None*, *protect_content: Optional[bool] = None*, *message_thread_id: Optional[int] = None*) → *Message*

Use this method to send audio files, if you want Telegram clients to display the file as a playable voice message. For this to work, your audio must be in an .OGG file encoded with OPUS (other formats may be sent as Audio or Document). On success, the sent Message is returned. Bots can currently send voice messages of up to 50 MB in size, this limit may be changed in the future.

Telegram documentation: https://core.telegram.org/bots/api#sendvoice

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `voice` (`str` or *telebot.types.InputFile*) – Audio file to send. Pass a file_id as String to send a file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a file from the Internet, or upload a new one using multipart/form-data.
- `caption` (`str`) – Voice message caption, 0-1024 characters after entities parsing
- `duration` (`int`) – Duration of the voice message in seconds
- `reply_to_message_id` (`int`) – If the message is a reply, ID of the original message
- `reply_markup` (*telebot.types.InlineKeyboardMarkup* or *telebot.types.ReplyKeyboardMarkup* or *telebot.types.ReplyKeyboardRemove* or *telebot.types.ForceReply*) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user.
- `parse_mode` (`str`) – Mode for parsing entities in the voice message caption. See formatting options for more details.
- `disable_notification` (`bool`) – Sends the message silently. Users will receive a notification with no sound.

- `timeout` (`int`) – Timeout in seconds for the request.
- `caption_entities` (`list` of *telebot.types.MessageEntity*) – A JSON-serialized list of special entities that appear in the caption, which can be specified instead of parse_mode
- `allow_sending_without_reply` (`bool`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- `protect_content` (`bool`) – Protects the contents of the sent message from forwarding and saving
- `message_thread_id` (`int`) – Identifier of a message thread, in which the message will be sent

Результат
On success, the sent Message is returned.

`async set_chat_administrator_custom_title(`*chat_id: Union[int, str]*, *user_id: int*, *custom_title: str*`)` → bool

Use this method to set a custom title for an administrator in a supergroup promoted by the bot. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#setchatadministratorcustomtitle

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target supergroup (in the format @supergroupusername)
- `user_id` (`int`) – Unique identifier of the target user
- `custom_title` (`str`) – New custom title for the administrator; 0-16 characters, emoji are not allowed

Результат
True on success.

Тип результата
`bool`

`async set_chat_description(`*chat_id: Union[int, str]*, *description: Optional[str] = None*`)` → bool

Use this method to change the description of a supergroup or a channel. The bot must be an administrator in the chat for this to work and must have the appropriate admin rights.

Telegram documentation: https://core.telegram.org/bots/api#setchatdescription

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `description` (`str`) – Str: New chat description, 0-255 characters

Результат
True on success.

Тип результата
`bool`

`async set_chat_menu_button(`*chat_id: Optional[Union[int, str]] = None*, *menu_button: Optional[*MenuButton*] = None*`)` → bool

Use this method to change the bot's menu button in a private chat, or the default menu button. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#setchatmenubutton

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target private chat. If not specified, default bot's menu button will be changed.
- `menu_button` (*telebot.types.MenuButton*) – A JSON-serialized object for the new bot's menu button. Defaults to MenuButtonDefault

Результат
True on success.

Тип результата
`bool`

`async set_chat_permissions(`*chat_id: Union[int, str]*, *permissions:* ChatPermissions`)` → bool

Use this method to set default chat permissions for all members. The bot must be an administrator in the group or a supergroup for this to work and must have the can_restrict_members admin rights.

Telegram documentation: https://core.telegram.org/bots/api#setchatpermissions

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target supergroup (in the format @supergroupusername)
- `permissions` (`telebot.types..ChatPermissions`) – New default chat permissions

Результат
True on success

Тип результата
`bool`

`async set_chat_photo(`*chat_id: Union[int, str]*, *photo: Any*`)` → bool

Use this method to set a new profile photo for the chat. Photos can't be changed for private chats. The bot must be an administrator in the chat for this to work and must have the appropriate admin rights. Returns True on success. Note: In regular groups (non-supergroups), this method will only work if the 'All Members Are Admins' setting is off in the target group.

Telegram documentation: https://core.telegram.org/bots/api#setchatphoto

Параметры

- `chat_id` (`int` or `str`) – Int or Str: Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `photo` (`typing.Union[file_like, str]`) – InputFile: New chat photo, uploaded using multipart/form-data

Результат
True on success.

Тип результата
`bool`

`async set_chat_sticker_set(`*chat_id: Union[int, str]*, *sticker_set_name: str*`)` → *StickerSet*

Use this method to set a new group sticker set for a supergroup. The bot must be an administrator in the chat for this to work and must have the appropriate administrator rights. Use the field

can_set_sticker_set optionally returned in getChat requests to check if the bot can use this method. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#setchatstickerset

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target supergroup (in the format @supergroupusername)
- `sticker_set_name` (`str`) – Name of the sticker set to be set as the group sticker set

Результат
StickerSet object

Тип результата
telebot.types.StickerSet

`async set_chat_title(`*chat_id: Union[int, str]*, *title: str*`)` → bool

Use this method to change the title of a chat. Titles can't be changed for private chats. The bot must be an administrator in the chat for this to work and must have the appropriate admin rights. Returns True on success. Note: In regular groups (non-supergroups), this method will only work if the 'All Members Are Admins' setting is off in the target group.

Telegram documentation: https://core.telegram.org/bots/api#setchattitle

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `title` (`str`) – New chat title, 1-255 characters

Результат
True on success.

Тип результата
`bool`

`async set_game_score(`*user_id: Union[int, str]*, *score: int*, *force: Optional[bool] = None*, *chat_id: Optional[Union[int, str]] = None*, *message_id: Optional[int] = None*, *inline_message_id: Optional[str] = None*, *disable_edit_message: Optional[bool] = None*`)` → Union[*Message*, bool]

Sets the value of points in the game to a specific user.

Telegram documentation: https://core.telegram.org/bots/api#setgamescore

Параметры

- `user_id` (`int` or `str`) – User identifier
- `score` (`int`) – New score, must be non-negative
- `force` (`bool`) – Pass True, if the high score is allowed to decrease. This can be useful when fixing mistakes or banning cheaters
- `chat_id` (`int` or `str`) – Required if inline_message_id is not specified. Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `message_id` (`int`) – Required if inline_message_id is not specified. Identifier of the sent message

- `inline_message_id` (`str`) – Required if chat_id and message_id are not specified. Identifier of the inline message
- `disable_edit_message` (`bool`) – Pass True, if the game message should not be automatically edited to include the current scoreboard

Результат
On success, if the message was sent by the bot, returns the edited Message, otherwise returns True.

Тип результата
`types.Message` or `bool`

`async set_my_commands(`*commands: List[*BotCommand*]*, *scope: Optional[*BotCommandScope*]* = *None*, *language_code: Optional[str]* = *None*`)` → bool

Use this method to change the list of the bot's commands.

Telegram documentation: https://core.telegram.org/bots/api#setmycommands

Параметры

- `commands` (`list` of *`telebot.types.BotCommand`*) – List of BotCommand. At most 100 commands can be specified.
- `scope` (*`telebot.types.BotCommandScope`*) – The scope of users for which the commands are relevant. Defaults to BotCommandScopeDefault.
- `language_code` (`str`) – A two-letter ISO 639-1 language code. If empty, commands will be applied to all users from the given scope, for whose language there are no dedicated commands

Результат
True on success.

Тип результата
`bool`

`async set_my_default_administrator_rights(`*rights: Optional[*ChatAdministratorRights*]* = *None*, *for_channels: Optional[bool]* = *None*`)` → bool

Use this method to change the default administrator rights requested by the bot when it's added as an administrator to groups or channels. These rights will be suggested to users, but they are are free to modify the list before adding the bot. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#setmydefaultadministratorrights

Параметры

- `rights` (*`telebot.types.ChatAdministratorRights`*) – A JSON-serialized object describing new default administrator rights. If not specified, the default administrator rights will be cleared.
- `for_channels` (`bool`) – Pass True to change the default administrator rights of the bot in channels. Otherwise, the default administrator rights of the bot for groups and supergroups will be changed.

Результат
True on success.

Тип результата
`bool`

async set_state(*user_id: int*, *state: Union[State, int, str]*, *chat_id: Optional[int] = None*)

Sets a new state of a user.

Примечание: You should set both user id and chat id in order to set state for a user in a chat. Otherwise, if you only set user_id, chat_id will equal to user_id, this means that state will be set for the user in his private chat with a bot.

Параметры

- `user_id` (`int`) – User's identifier
- `state` (`int` or `str` or `telebot.types.State`) – new state. can be string, integer, or `telebot.types.State`
- `chat_id` (`int`) – Chat's identifier

Результат

None

async set_sticker_position_in_set(*sticker: str*, *position: int*) → bool

Use this method to move a sticker in a set created by the bot to a specific position . Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#setstickerpositioninset

Параметры

- `sticker` (`str`) – File identifier of the sticker
- `position` (`int`) – New sticker position in the set, zero-based

Результат

On success, True is returned.

Тип результата

`bool`

async set_sticker_set_thumb(*name: str*, *user_id: int*, *thumb: Optional[Union[str, Any]] = None*)

Use this method to set the thumbnail of a sticker set. Animated thumbnails can be set for animated sticker sets only. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#setstickersetthumb

Параметры

- `name` (`str`) – Sticker set name
- `user_id` (`int`) – User identifier
- `thumb` (`filelike object`) –

Результат

On success, True is returned.

Тип результата

`bool`

set_update_listener(*func: Awaitable*)

Update listener is a function that gets any update.

Параметры

`func` (`Awaitable`) – function that should get update.

Список 25: Example on asynchronous update listeners.

```
async def update_listener(new_messages):
    for message in new_messages:
        print(message.text) # Prints message text

bot.set_update_listener(update_listener)
```

Результат
None

async set_webhook(*url: Optional[str] = None*, *certificate: Optional[Union[str, Any]] = None*, *max_connections: Optional[int] = None*, *allowed_updates: Optional[List[str]] = None*, *ip_address: Optional[str] = None*, *drop_pending_updates: Optional[bool] = None*, *timeout: Optional[int] = None*, *secret_token: Optional[str] = None*) → bool

Use this method to specify a URL and receive incoming updates via an outgoing webhook. Whenever there is an update for the bot, we will send an HTTPS POST request to the specified URL, containing a JSON-serialized Update. In case of an unsuccessful request, we will give up after a reasonable amount of attempts. Returns True on success.

If you'd like to make sure that the webhook was set by you, you can specify secret data in the parameter secret_token. If specified, the request will contain a header "X-Telegram-Bot-Api-Secret-Token" with the secret token as content.

Telegram Documentation: https://core.telegram.org/bots/api#setwebhook

Параметры

- `url` (`str`, optional) – HTTPS URL to send updates to. Use an empty string to remove webhook integration, defaults to None
- `certificate` (`str`, optional) – Upload your public key certificate so that the root certificate in use can be checked, defaults to None
- `max_connections` (`int`, optional) – The maximum allowed number of simultaneous HTTPS connections to the webhook for update delivery, 1-100. Defaults to 40. Use lower values to limit the load on your bot's server, and higher values to increase your bot's throughput, defaults to None
- `allowed_updates` (`list`, optional) – A JSON-serialized list of the update types you want your bot to receive. For example, specify ["message", "edited_channel_post", "callback_query"] to only receive updates of these types. See Update for a complete list of available update types. Specify an empty list to receive all update types except chat_member (default). If not specified, the previous setting will be used.

  Please note that this parameter doesn't affect updates created before the call to the setWebhook, so unwanted updates may be received for a short period of time. Defaults to None
- `ip_address` (`str`, optional) – The fixed IP address which will be used to send webhook requests instead of the IP address resolved through DNS, defaults to None
- `drop_pending_updates` (`bool`, optional) – Pass True to drop all pending updates, defaults to None
- `timeout` (`int`, optional) – Timeout of a request, defaults to None

- `secret_token` (`str`, optional) – A secret token to be sent in a header "X-Telegram-Bot-Api-Secret-Token" in every webhook request, 1-256 characters. Only characters A-Z, a-z, 0-9, _ and - are allowed. The header is useful to ensure that the request comes from a webhook set by you. Defaults to None

Результат
True on success.

Тип результата
`bool` if the request was successful.

`setup_middleware(`*middleware:* BaseMiddleware`)`

Setup middleware.

Примечание: Take a look at the *telebot.asyncio_handler_backends.BaseMiddleware* section for more.

Параметры
`middleware` (*telebot.asyncio_handler_backends.BaseMiddleware*) – Middleware-class.

Результат
None

`shipping_query_handler(`*func*, ***kwargs*`)`

Handles new incoming shipping query. Only for invoices with flexible price. As a parameter to the decorator function, it passes *telebot.types.ShippingQuery* object.

Параметры
- `func` (`function`) – Function executed as a filter
- `kwargs` – Optional keyword arguments(custom filters)

Результат
None

`async skip_updates()`

Skip existing updates. Only last update will remain on server.

`async stop_message_live_location(`*chat_id: Optional[Union[int, str]] = None*, *message_id: Optional[int] = None*, *inline_message_id: Optional[str] = None*, *reply_markup: Optional[Union[*InlineKeyboardMarkup*,* ReplyKeyboardMarkup*,* ReplyKeyboardRemove*,* ForceReply*]] = None*, *timeout: Optional[int] = None*`) →` *Message*

Use this method to stop updating a live location message before live_period expires. On success, if the message is not an inline message, the edited Message is returned, otherwise True is returned.

Telegram documentation: https://core.telegram.org/bots/api#stopmessagelivelocation

Параметры
- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `message_id` (`int`) – Required if inline_message_id is not specified. Identifier of the message with live location to stop

- `inline_message_id` (`str`) – Required if chat_id and message_id are not specified. Identifier of the inline message with live location to stop
- `reply_markup` (*telebot.types.InlineKeyboardMarkup* or *telebot.types.ReplyKeyboardMarkup* or *telebot.types.ReplyKeyboardRemove* or *telebot.types.ForceReply*) – A JSON-serialized object for a new inline keyboard.
- `timeout` (`int`) – Timeout in seconds for the request.

Результат
On success, if the message is not an inline message, the edited Message is returned, otherwise True is returned.

Тип результата
telebot.types.Message or bool

async stop_poll(*chat_id: Union[int, str]*, *message_id: int*, *reply_markup: Optional[Union[*InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply*]] = None*) → *Poll*

Use this method to stop a poll which was sent by the bot. On success, the stopped Poll is returned.

Telegram documentation: https://core.telegram.org/bots/api#stoppoll

Параметры

- `chat_id` (`int` | `str`) – Unique identifier for the target chat or username of the target channel
- `message_id` (`int`) – Identifier of the original message with the poll
- `reply_markup` (`InlineKeyboardMarkup` | `ReplyKeyboardMarkup` | `ReplyKeyboardRemove` | `ForceReply`) – A JSON-serialized object for a new message markup.

Результат
On success, the stopped Poll is returned.

Тип результата
`types.Poll`

async unban_chat_member(*chat_id: Union[int, str]*, *user_id: int*, *only_if_banned: Optional[bool] = False*) → bool

Use this method to unban a previously kicked user in a supergroup 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.

Telegram documentation: https://core.telegram.org/bots/api#unbanchatmember

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target group or username of the target supergroup or channel (in the format @username)
- `user_id` (`int`) – Unique identifier of the target user
- `only_if_banned` (`bool`) – Do nothing if the user is not banned

Результат
True on success

Тип результата
`bool`

`async unban_chat_sender_chat(`*chat_id: Union[int, str]*, *sender_chat_id: Union[int, str]*`)` → bool

Use this method to unban a previously banned channel chat in a supergroup or channel. The bot must be an administrator for this to work and must have the appropriate administrator rights. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#unbanchatsenderchat

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `sender_chat_id` (`int` or `str`) – Unique identifier of the target sender chat.

Результат
True on success.

Тип результата
`bool`

`async unpin_all_chat_messages(`*chat_id: Union[int, str]*`)` → bool

Use this method to unpin a all pinned messages in a supergroup chat. The bot must be an administrator in the chat for this to work and must have the appropriate admin rights. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#unpinallchatmessages

Параметры
`chat_id` (`int` or `str`) – Int or Str: Unique identifier for the target chat or username of the target channel (in the format @channelusername)

Результат
True on success.

Тип результата
`bool`

`async unpin_all_forum_topic_messages(`*chat_id: Union[str, int]*, *message_thread_id: int*`)` → bool

Use this method to clear the list of pinned messages in a forum topic. The bot must be an administrator in the chat for this to work and must have the can_pin_messages administrator right in the supergroup. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#unpinallforumtopicmessages

Параметры

- `chat_id` (`int` or `str`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- `message_thread_id` (`int`) – Identifier of the topic

Результат
On success, True is returned.

Тип результата
`bool`

async unpin_chat_message(*chat_id: Union[int, str]*, *message_id: Optional[int] = None*) → bool

Use this method to unpin specific pinned message in a supergroup chat. The bot must be an administrator in the chat for this to work and must have the appropriate admin rights. Returns True on success.

Telegram documentation: https://core.telegram.org/bots/api#unpinchatmessage

Параметры

- chat_id (int or str) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- message_id (int) – Int: Identifier of a message to unpin

Результат

True on success.

Тип результата

bool

async upload_sticker_file(*user_id: int*, *png_sticker: Union[Any, str]*) → *File*

Use this method to upload a .png file with a sticker for later use in createNewStickerSet and addStickerToSet methods (can be used multiple times). Returns the uploaded File on success.

Telegram documentation: https://core.telegram.org/bots/api#uploadstickerfile

Параметры

- user_id (int) – User identifier of sticker set owner
- png_sticker (filelike object) – PNG image with the sticker, must be up to 512 kilobytes in size, dimensions must not exceed 512px, and either width or height must be exactly 512px.

Результат

On success, the sent file is returned.

Тип результата

telebot.types.File

property user

class telebot.async_telebot.ExceptionHandler

Базовые классы: object

Class for handling exceptions while Polling

handle(*exception*)

class telebot.async_telebot.Handler(*callback*, **args*, ***kwargs*)

Базовые классы: object

Class for (next step|reply) handlers

Asyncio filters

class telebot.asyncio_filters.AdvancedCustomFilter

Базовые классы: ABC

Advanced Custom Filter base class. Create child class with check() method. Accepts two parameters, returns bool: True - filter passed, False - filter failed. message: Message class text: Filter value given in handler

Child classes should have .key property.

Список 26: Example on creating an advanced custom filter.

```
class TextStartsFilter(AdvancedCustomFilter):
    # Filter to check whether message starts with some text.
    key = 'text_startswith'

    def check(self, message, text):
        return message.text.startswith(text)
```

async check(*message*, *text*)

Perform a check.

key: str = None

class telebot.asyncio_filters.ChatFilter

Базовые классы: *AdvancedCustomFilter*

Check whether chat_id corresponds to given chat_id.

Список 27: Example on using this filter:

```
@bot.message_handler(chat_id=[99999])
# your function
```

key: str = 'chat_id'

class telebot.asyncio_filters.ForwardFilter

Базовые классы: *SimpleCustomFilter*

Check whether message was forwarded from channel or group.

Список 28: Example on using this filter:

```
@bot.message_handler(is_forwarded=True)
# your function
```

key: str = 'is_forwarded'

class telebot.asyncio_filters.IsAdminFilter(*bot*)

Базовые классы: *SimpleCustomFilter*

Check whether the user is administrator / owner of the chat.

Список 29: Example on using this filter:

```
@bot.message_handler(chat_types=['supergroup'], is_chat_admin=True)
# your function
```

key: str = 'is_chat_admin'

class telebot.asyncio_filters.IsDigitFilter

Базовые классы: *SimpleCustomFilter*

Filter to check whether the string is made up of only digits.

Список 30: Example on using this filter:

```
@bot.message_handler(is_digit=True)
# your function
```

key: str = 'is_digit'

class telebot.asyncio_filters.IsReplyFilter

Базовые классы: *SimpleCustomFilter*

Check whether message is a reply.

Список 31: Example on using this filter:

```
@bot.message_handler(is_reply=True)
# your function
```

key: str = 'is_reply'

class telebot.asyncio_filters.LanguageFilter

Базовые классы: *AdvancedCustomFilter*

Check users language_code.

Список 32: Example on using this filter:

```
@bot.message_handler(language_code=['ru'])
# your function
```

key: str = 'language_code'

class telebot.asyncio_filters.SimpleCustomFilter

Базовые классы: ABC

Simple Custom Filter base class. Create child class with check() method. Accepts only message, returns bool value, that is compared with given in handler.

Child classes should have .key property.

Список 33: Example on creating a simple custom filter.

```
class ForwardFilter(SimpleCustomFilter):
    # Check whether message was forwarded from channel or group.
    key = 'is_forwarded'
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    def check(self, message):
        return message.forward_date is not None
```

async check(*message*)

Perform a check.

key: str = None

class telebot.asyncio_filters.StateFilter(*bot*)

Базовые классы: *AdvancedCustomFilter*

Filter to check state.

Список 34: Example on using this filter:

```
@bot.message_handler(state=1)
# your function
```

key: str = 'state'

class telebot.asyncio_filters.TextContainsFilter

Базовые классы: *AdvancedCustomFilter*

Filter to check Text message. key: text

Список 35: Example on using this filter:

```
# Will respond if any message.text contains word 'account'
@bot.message_handler(text_contains=['account'])
# your function
```

key: str = 'text_contains'

class telebot.asyncio_filters.TextFilter(*equals: Optional[str] = None, contains: Optional[Union[list, tuple]] = None, starts_with: Optional[Union[str, list, tuple]] = None, ends_with: Optional[Union[str, list, tuple]] = None, ignore_case: bool = False*)

Базовые классы: object

Advanced text filter to check (types.Message, types.CallbackQuery, types.InlineQuery, types.Poll)

example of usage is in examples/asynchronous_telebot/custom_filters/advanced_text_filter.py

Параметры

- equals (str) – string, True if object's text is equal to passed string
- contains (*list[str] or tuple[str]*) – list[str] or tuple[str], True if any string element of iterable is in text
- starts_with (str) – string, True if object's text starts with passed string
- ends_with (str) – string, True if object's text starts with passed string
- ignore_case (bool) – bool (default False), case insensitive

Исключение

ValueError – if incorrect value for a parameter was supplied

Результат
None

class telebot.asyncio_filters.TextMatchFilter

Базовые классы: *AdvancedCustomFilter*

Filter to check Text message.

Список 36: Example on using this filter:

```
@bot.message_handler(text=['account'])
# your function
```

key: str = 'text'

class telebot.asyncio_filters.TextStartsFilter

Базовые классы: *AdvancedCustomFilter*

Filter to check whether message starts with some text.

Список 37: Example on using this filter:

```
# Will work if message.text starts with 'sir'.
@bot.message_handler(text_startswith='sir')
# your function
```

key: str = 'text_startswith'

Asyncio handler backends

File with all middleware classes, states.

class telebot.asyncio_handler_backends.BaseMiddleware

Базовые классы: object

Base class for middleware. Your middlewares should be inherited from this class.

Set update_sensitive=True if you want to get different updates on different functions. For example, if you want to handle pre_process for message update, then you will have to create pre_process_message function, and so on. Same applies to post_process.

Список 38: Example of class-based middlewares

```
class MyMiddleware(BaseMiddleware):
    def __init__(self):
        self.update_sensitive = True
        self.update_types = ['message', 'edited_message']

    async def pre_process_message(self, message, data):
        # only message update here
        pass

    async def post_process_message(self, message, data, exception):
        pass # only message update here for post_process

    async def pre_process_edited_message(self, message, data):
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        # only edited_message update here
        pass

    async def post_process_edited_message(self, message, data, exception):
        pass # only edited_message update here for post_process
```

async post_process(*message*, *data*, *exception*)

async pre_process(*message*, *data*)

update_sensitive: bool = False

class telebot.asyncio_handler_backends.CancelUpdate

Базовые классы: object

Class for canceling updates. Just return instance of this class in middleware to skip update. Update will skip handler and execution of post_process in middlewares.

class telebot.asyncio_handler_backends.ContinueHandling

Базовые классы: object

Class for continue updates in handlers. Just return instance of this class in handlers to continue process.

Список 39: Example of using ContinueHandling

```
@bot.message_handler(commands=['start'])
async def start(message):
    await bot.send_message(message.chat.id, 'Hello World!')
    return ContinueHandling()

@bot.message_handler(commands=['start'])
async def start2(message):
    await bot.send_message(message.chat.id, 'Hello World2!')
```

class telebot.asyncio_handler_backends.SkipHandler

Базовые классы: object

Class for skipping handlers. Just return instance of this class in middleware to skip handler. Update will go to post_process, but will skip execution of handler.

class telebot.asyncio_handler_backends.State

Базовые классы: object

Class representing a state.

```
class MyStates(StatesGroup):
    my_state = State() # returns my_state:State string.
```

class telebot.asyncio_handler_backends.StatesGroup

Базовые классы: object

Class representing common states.

```
class MyStates(StatesGroup):
    my_state = State() # returns my_state:State string.
```

Extensions

1.3.6 Callback data factory

callback_data file

Callback data factory's file.

class telebot.callback_data.CallbackData(*parts, prefix: str, sep=':')

Базовые классы: object

Callback data factory This class will help you to work with CallbackQuery

filter(**config) → CallbackDataFilter

Generate filter

Параметры
config – specified named parameters will be checked with CallbackQuery.data

Результат
CallbackDataFilter class

new(*args, **kwargs) → str

Generate callback data

Параметры

- args – positional parameters of CallbackData instance parts
- kwargs – named parameters

Результат
str

parse(callback_data: str) → Dict[str, str]

Parse data from the callback data

Параметры
callback_data – string, use to telebot.types.CallbackQuery to parse it from string to a dict

Результат
dict parsed from callback data

class telebot.callback_data.CallbackDataFilter(factory, config: Dict[str, str])

Базовые классы: object

Filter for CallbackData.

check(query) → bool

Checks if query.data appropriates to specified config

Параметры
query (telebot.types.CallbackQuery) – telebot.types.CallbackQuery

Результат
True if query.data appropriates to specified config

Тип результата
bool

1.3.7 Utils

util file

telebot.util.antiflood(*function: Callable*, **args*, ***kwargs*)

Use this function inside loops in order to avoid getting TooManyRequests error. Example:

```
from telebot.util import antiflood
for chat_id in chat_id_list:
msg = antiflood(bot.send_message, chat_id, text)
```

Параметры

- `function` (`Callable`) – The function to call
- `args` (`tuple`) – The arguments to pass to the function
- `kwargs` (`dict`) – The keyword arguments to pass to the function

Результат

None

telebot.util.chunks(*lst*, *n*)

Yield successive n-sized chunks from lst.

telebot.util.content_type_media = ['text', 'audio', 'document', 'animation', 'game', 'photo', 'sticker', 'video', 'video_note', 'voice', 'contact', 'location', 'venue', 'dice', 'invoice', 'successful_payment', 'connected_website', 'poll', 'passport_data', 'web_app_data']

Contains all media content types.

telebot.util.content_type_service = ['new_chat_members', 'left_chat_member', 'new_chat_title', 'new_chat_photo', 'delete_chat_photo', 'group_chat_created', 'supergroup_chat_created', 'channel_chat_created', 'migrate_to_chat_id', 'migrate_from_chat_id', 'pinned_message', 'proximity_alert_triggered', 'video_chat_scheduled', 'video_chat_started', 'video_chat_ended', 'video_chat_participants_invited', 'message_auto_delete_timer_changed', 'forum_topic_created', 'forum_topic_closed', 'forum_topic_reopened']

Contains all service content types such as *User joined the group.*

telebot.util.escape(*text: str*) → str

Replaces the following chars in *text* ("&" with „&", „<" with „<" and „>" with „>").

Параметры

`text` – the text to escape

Результат

the escaped text

telebot.util.extract_arguments(*text: str*) → str

Returns the argument after the command.

Список 40: Examples:

```
extract_arguments("/get name"): 'name'
extract_arguments("/get"): ''
extract_arguments("/get@botName name"): 'name'
```

Параметры
text (str) – String to extract the arguments from a command

Результат
the arguments if *text* is a command (according to is_command), else None.

Тип результата
str or None

telebot.util.extract_command(*text: str*) → Optional[str]

Extracts the command from *text* (minus the „/“) if *text* is a command (see is_command). If *text* is not a command, this function returns None.

Список 41: Examples:

```
extract_command('/help'): 'help'
extract_command('/help@BotName'): 'help'
extract_command('/search black eyed peas'): 'search'
extract_command('Good day to you'): None
```

Параметры
text (str) – String to extract the command from

Результат
the command if *text* is a command (according to is_command), else None.

Тип результата
str or None

telebot.util.generate_random_token() → str

Generates a random token consisting of letters and digits, 16 characters long.

Результат
a random token

Тип результата
str

telebot.util.is_bytes(*var*) → bool

Returns True if the given object is a bytes object.

Параметры
var (object) – object to be checked

Результат
True if the given object is a bytes object.

Тип результата
bool

telebot.util.is_command(*text: str*) → bool

Checks if *text* is a command. Telegram chat commands start with the „/“ character.

Параметры
`text` (`str`) – Text to check.

Результат
True if *text* is a command, else False.

Тип результата
`bool`

telebot.util.is_dict(*var*) → bool

Returns True if the given object is a dictionary.

Параметры
`var` (`object`) – object to be checked

Результат
True if the given object is a dictionary.

Тип результата
`bool`

telebot.util.is_pil_image(*var*) → bool

Returns True if the given object is a PIL.Image.Image object.

Параметры
`var` (`object`) – object to be checked

Результат
True if the given object is a PIL.Image.Image object.

Тип результата
`bool`

telebot.util.is_string(*var*) → bool

Returns True if the given object is a string.

telebot.util.parse_web_app_data(*token: str*, *raw_ init_ data: str*)

Parses web app data.

Параметры

- `token` (`str`) – The bot token
- `raw_init_data` (`str`) – The raw init data

Результат
The parsed init data

telebot.util.pil_image_to_file(*image*, *extension='JPEG'*, *quality='web_ low'*)

telebot.util.quick_markup(*values: Dict[str, Dict[str, Any]]*, *row_ width: int = 2*) → *InlineKeyboardMarkup*

Returns a reply markup from a dict in this format: {„text“: kwargs} This is useful to avoid always typing „btn1 = InlineKeyboardButton(...)“ „btn2 = InlineKeyboardButton(...)“

Example:

Список 42: Using quick_markup:

```
quick_markup({
    'Twitter': {'url': 'https://twitter.com'},
    'Facebook': {'url': 'https://facebook.com'},
    'Back': {'callback_data': 'whatever'}
}, row_width=2):
# returns an InlineKeyboardMarkup with two buttons in a row, one leading to Twitter,
↪ the other to facebook
# and a back button below

# kwargs can be:
{
    'url': None,
    'callback_data': None,
    'switch_inline_query': None,
    'switch_inline_query_current_chat': None,
    'callback_game': None,
    'pay': None,
    'login_url': None,
    'web_app': None
}
```

Параметры

- `values` (`dict`) – a dict containing all buttons to create in this format: {text: kwargs} {str:}
- `row_width` (`int`) – int row width

Результат
InlineKeyboardMarkup

Тип результата
`types.InlineKeyboardMarkup`

`telebot.util.smart_split`(*text: str*, *chars_per_string: int = 4096*) → List[str]

Splits one string into multiple strings, with a maximum amount of *chars_per_string* characters per string. This is very useful for splitting one giant message into multiples. If *chars_per_string* > 4096: *chars_per_string* = 4096. Splits by „n“, „. „ or „ „ in exactly this priority.

Параметры

- `text` (`str`) – The text to split
- `chars_per_string` (`int`) – The number of maximum characters per part the text is split to.

Результат
The splitted text as a list of strings.

Тип результата
`list` of `str`

`telebot.util.split_string`(*text: str*, *chars_per_string: int*) → List[str]

Splits one string into multiple strings, with a maximum amount of *chars_per_string* characters per string. This is very useful for splitting one giant message into multiples.

Параметры

- `text` (`str`) – The text to split
- `chars_per_string` (`int`) – The number of characters per line the text is split into.

Результат
The splitted text as a list of strings.

Тип результата
`list` of `str`

```
telebot.util.update_types = ['message', 'edited_message', 'channel_post',
'edited_channel_post', 'inline_query', 'chosen_inline_result', 'callback_query',
'shipping_query', 'pre_checkout_query', 'poll', 'poll_answer', 'my_chat_member',
'chat_member', 'chat_join_request']
```

All update types, should be used for allowed_updates parameter in polling.

`telebot.util.user_link(`*user: User*, *include_id: bool = False*`)` → str

Returns an HTML user link. This is useful for reports. Attention: Don't forget to set parse_mode to „HTML"!

Список 43: Example:

```
bot.send_message(your_user_id, user_link(message.from_user) + ' started the bot!',␣
↪parse_mode='HTML')
```

Примечание: You can use formatting.* for all other formatting options(bold, italic, links, and etc.) This method is kept for backward compatibility, and it is recommended to use formatting.* for more options.

Параметры

- `user` (*`telebot.types.User`*) – the user (not the user_id)
- `include_id` (`bool`) – include the user_id

Результат
HTML user link

Тип результата
`str`

`telebot.util.validate_web_app_data(`*token: str*, *raw_init_data: str*`)`

Validates web app data.

Параметры

- `token` (`str`) – The bot token
- `raw_init_data` (`str`) – The raw init data

Результат
The parsed init data

`telebot.util.webhook_google_functions(`*bot*, *request*`)`

A webhook endpoint for Google Cloud Functions FaaS.

Параметры

- bot (*telebot.TeleBot* or *telebot.async_telebot.AsyncTeleBot*) – The bot instance
- request (flask.Request) – The request object

Результат
The response object

1.3.8 Formatting options

Markdown & HTML formatting functions.

Добавлено в версии 4.5.1.

telebot.formatting.escape_html(*content: str*) → str

Escapes HTML characters in a string of HTML.

Параметры
content (str) – The string of HTML to escape.

Результат
The escaped string.

Тип результата
str

telebot.formatting.escape_markdown(*content: str*) → str

Escapes Markdown characters in a string of Markdown.

Credits to: simonsmh

Параметры
content (str) – The string of Markdown to escape.

Результат
The escaped string.

Тип результата
str

telebot.formatting.format_text(**args*, *separator='\n'*)

Formats a list of strings into a single string.

```
format_text( # just an example
    mbold('Hello'),
    mitalic('World')
)
```

Параметры

- args (str) – Strings to format.
- separator (str) – The separator to use between each string.

Результат
The formatted string.

Тип результата
str

telebot.formatting.hbold(*content: str*, *escape: Optional[bool] = True*) → str

Returns an HTML-formatted bold string.

Параметры

- content (str) – The string to bold.
- escape (bool) – True if you need to escape special characters. Defaults to True.

Результат

The formatted string.

Тип результата

str

telebot.formatting.hcode(*content: str*, *escape: Optional[bool] = True*) → str

Returns an HTML-formatted code string.

Параметры

- content (str) – The string to code.
- escape (bool) – True if you need to escape special characters. Defaults to True.

Результат

The formatted string.

Тип результата

str

telebot.formatting.hide_link(*url: str*) → str

Hide url of an image.

Параметры

url (str) – The url of the image.

Результат

The hidden url.

Тип результата

str

telebot.formatting.hitalic(*content: str*, *escape: Optional[bool] = True*) → str

Returns an HTML-formatted italic string.

Параметры

- content (str) – The string to italicize.
- escape (bool) – True if you need to escape special characters. Defaults to True.

Результат

The formatted string.

Тип результата

str

telebot.formatting.hlink(*content: str*, *url: str*, *escape: Optional[bool] = True*) → str

Returns an HTML-formatted link string.

Параметры

- content (str) – The string to link.
- url (str) – The URL to link to.

- escape (bool) – True if you need to escape special characters. Defaults to True.

Результат
The formatted string.

Тип результата
str

telebot.formatting.hpre(*content: str*, *escape: Optional[bool] = True*, *language: str = ''*) → str

Returns an HTML-formatted preformatted string.

Параметры

- content (str) – The string to preformatted.
- escape (bool) – True if you need to escape special characters. Defaults to True.

Результат
The formatted string.

Тип результата
str

telebot.formatting.hspoiler(*content: str*, *escape: Optional[bool] = True*) → str

Returns an HTML-formatted spoiler string.

Параметры

- content (str) – The string to spoiler.
- escape (bool) – True if you need to escape special characters. Defaults to True.

Результат
The formatted string.

Тип результата
str

telebot.formatting.hstrikethrough(*content: str*, *escape: Optional[bool] = True*) → str

Returns an HTML-formatted strikethrough string.

Параметры

- content (str) – The string to strikethrough.
- escape (bool) – True if you need to escape special characters. Defaults to True.

Результат
The formatted string.

Тип результата
str

telebot.formatting.hunderline(*content: str*, *escape: Optional[bool] = True*) → str

Returns an HTML-formatted underline string.

Параметры

- content (str) – The string to underline.
- escape (bool) – True if you need to escape special characters. Defaults to True.

Результат
The formatted string.

Тип результата
str

telebot.formatting.mbold(*content: str*, *escape: Optional[bool] = True*) → str

Returns a Markdown-formatted bold string.

Параметры

- content (str) – The string to bold.
- escape (bool) – True if you need to escape special characters. Defaults to True.

Результат
The formatted string.

Тип результата
str

telebot.formatting.mcode(*content: str*, *language: str = ''*, *escape: Optional[bool] = True*) → str

Returns a Markdown-formatted code string.

Параметры

- content (str) – The string to code.
- escape (bool) – True if you need to escape special characters. Defaults to True.

Результат
The formatted string.

Тип результата
str

telebot.formatting.mitalic(*content: str*, *escape: Optional[bool] = True*) → str

Returns a Markdown-formatted italic string.

Параметры

- content (str) – The string to italicize.
- escape (bool) – True if you need to escape special characters. Defaults to True.

Результат
The formatted string.

Тип результата
str

telebot.formatting.mlink(*content: str*, *url: str*, *escape: Optional[bool] = True*) → str

Returns a Markdown-formatted link string.

Параметры

- content (str) – The string to link.
- url (*str*) – The URL to link to.
- escape (bool) – True if you need to escape special characters. Defaults to True.

Результат
The formatted string.

Тип результата
str

telebot.formatting.mspoiler(*content: str*, *escape: Optional[bool] = True*) → str

Returns a Markdown-formatted spoiler string.

Параметры

- content (str) – The string to spoiler.
- escape (bool) – True if you need to escape special characters. Defaults to True.

Результат

The formatted string.

Тип результата

str

telebot.formatting.mstrikethrough(*content: str*, *escape: Optional[bool] = True*) → str

Returns a Markdown-formatted strikethrough string.

Параметры

- content (str) – The string to strikethrough.
- escape (bool) – True if you need to escape special characters. Defaults to True.

Результат

The formatted string.

Тип результата

str

telebot.formatting.munderline(*content: str*, *escape: Optional[bool] = True*) → str

Returns a Markdown-formatted underline string.

Параметры

- content (str) – The string to underline.
- escape (bool) – True if you need to escape special characters. Defaults to True.

Результат

The formatted string.

Тип результата

str

Глава 2

Indices and tables

- genindex
- modindex
- search

t

Символы

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W