pyTelegramBotAPI

Выпуск 4.17.0

coder2020official

апр. 25, 2024

Оглавление

1 TeleBot **1**
1.1 Чаты 1
1.2 Некоторые особенности: 1
1.3 Содержимое 2

2 Ссылки **297**

Содержание модулей Python **299**

Алфавитный указатель **301**

Глава 1

TeleBot

TeleBot это синхронная и асинхронная реализация Telegram Bot API.

1.1 Чаты

Англоязычный чат: Private chat

Русскоязычный чат: @pytelegrambotapi_talks_ru

Новости: @pyTelegramBotAPI

Pypi: Pypi

Исходники: Github repository

1.2 Некоторые особенности:

Простой в изучении и использовании.

Простой в понимании.

И синхронный, и асинхронный.

Примеры возможностей.

Состояния (стейты, FSM)

И другое...

1.3 Содержимое

1.3.1 Гайд по установке

Используя PIP

```
$ pip install pyTelegramBotAPI
```

Используя pipenv

```
$ pipenv install pyTelegramBotAPI
```

Клонируя репозиторий

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

Напрямую используя pip

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

Рекомендуется использовать первый вариант.

Новые версии библиотеки имеют больше фич, улучшений и баг фиксов. Не забывайте обновляться вызывая:

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

1.3.2 Быстрый старт

Синхронный телебот

```
#!/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()
```

Асинхронный телебот

```
#!/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*, *thumbnail=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
- `thumbnail` (*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

property thumb

class telebot.types.Audio(*file_id*, *file_unique_id*, *duration*, *performer=None*, *title=None*, *file_name=None*, *mime_type=None*, *file_size=None*, *thumbnail=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.
- `thumbnail` (*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

`property thumb`

`class telebot.types.Birthdate(`*day*, *month*, *year=None*, ***kwargs*`)`

Базовые классы: *JsonDeserializable*

This object represents a user's birthdate.

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

Параметры

- `day` (`int`) – Day of the user's birth; 1-31
- `month` (`int`) – Month of the user's birth; 1-12
- `year` (`int`) – Optional. Year of the user's birth

Результат
Instance of the class

Тип результата
Birthdate

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

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.BotDescription(*description: str*, ***kwargs*)

Базовые классы: *JsonDeserializable*

This object represents a bot description.

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

Параметры
`description` (`str`) – Bot description

Результат
Instance of the class

Тип результата
telebot.types.BotDescription

class telebot.types.BotName(*name: str*, ***kwargs*)

Базовые классы: *JsonDeserializable*

This object represents a bot name.

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

Параметры
`name` (`str`) – The bot name

Результат
Instance of the class

Тип результата
BotName

class telebot.types.BotShortDescription(*short_description: str*, ***kwargs*)

Базовые классы: *JsonDeserializable*

This object represents a bot short description.

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

Параметры
`short_description` (`str`) – Bot short description

Результат
Instance of the class

Тип результата
telebot.types.BotShortDescription

class telebot.types.BusinessConnection(*id*, *user*, *user_chat_id*, *date*, *can_reply*, *is_enabled*, ***kwargs*)

Базовые классы: *JsonDeserializable*

This object describes the connection of the bot with a business account.

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

Параметры

- `id` (`str`) – Unique identifier of the business connection
- `user` (*User*) – Business account user that created the business connection
- `user_chat_id` (`int`) – Identifier of a private chat with the user who created the business connection

- `date` (`int`) – Date the connection was established in Unix time
- `can_reply` (`bool`) – True, if the bot can act on behalf of the business account in chats that were active in the last 24 hours
- `is_enabled` (`bool`) – True, if the connection is active

Результат
Instance of the class

Тип результата
BusinessConnection

`class telebot.types.BusinessIntro(`*title=None*, *message=None*, *sticker=None*, ***kwargs*`)`

Базовые классы: *JsonDeserializable*

This object represents a business intro.

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

Параметры

- `title` (`str`) – Optional. Title text of the business intro
- `message` (`str`) – Optional. Message text of the business intro
- `sticker` (*Sticker*) – Optional. Sticker of the business intro

Результат
Instance of the class

Тип результата
BusinessIntro

`class telebot.types.BusinessLocation(`*address*, *location=None*, ***kwargs*`)`

Базовые классы: *JsonDeserializable*

This object represents a business location.

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

Параметры

- `address` (`str`) – Address of the business
- `location` (*Location*) – Optional. Location of the business

Результат
Instance of the class

Тип результата
BusinessLocation

`class telebot.types.BusinessMessagesDeleted(`*business_connection_id*, *chat*, *message_ids*, ***kwargs*`)`

Базовые классы: *JsonDeserializable*

This object is received when messages are deleted from a connected business account.

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

Параметры

- `business_connection_id` (`str`) – Unique identifier of the business connection
- `chat` (*Chat*) – Information about a chat in the business account. The bot may not have access to the chat or the corresponding user.

- `message_ids` (`list` of `int`) – A JSON-serialized list of identifiers of deleted messages in the chat of the business account

Результат
Instance of the class

Тип результата
BusinessMessagesDeleted

`class telebot.types.BusinessOpeningHours(time_zone_name, opening_hours, **kwargs)`

Базовые классы: *JsonDeserializable*

This object represents business opening hours.

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

Параметры

- `time_zone_name` (`str`) – Unique name of the time zone for which the opening hours are defined
- `opening_hours` (`list` of *BusinessOpeningHoursInterval*) – List of time intervals describing business opening hours

Результат
Instance of the class

Тип результата
BusinessOpeningHours

`class telebot.types.BusinessOpeningHoursInterval(opening_minute, closing_minute, **kwargs)`

Базовые классы: *JsonDeserializable*

This object represents a business opening hours interval.

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

Параметры

- `opening_minute` (`int`) – The minute's sequence number in a week, starting on Monday, marking the start of the time interval during which the business is open; 0 - 7 24 60
- `closing_minute` (`int`) – The minute's sequence number in a week, starting on Monday, marking the end of the time interval during which the business is open; 0 - 8 24 60

Результат
Instance of the class

Тип результата
BusinessOpeningHoursInterval

`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* or *telebot.types.InaccessibleMessage*) – Optional. Message sent by the bot with the callback button that originated the query
- `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, has_hidden_members=None, has_aggressive_anti_spam_enabled=None, emoji_status_expiration_date=None, available_reactions=None, accent_color_id=None, background_custom_emoji_id=None, profile_accent_color_id=None, profile_background_custom_emoji_id=None, has_visible_history=None, unrestrict_boost_count=None, custom_emoji_sticker_set_name=None, business_intro=None, business_location=None, business_opening_hours=None, personal_chat=None, birthdate=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.
- `birthdate` (`str`) – Optional. Birthdate of the other party in a private chat. Returned only in getChat.
- `business_intro` (*telebot.types.BusinessIntro*) – Optional. Business intro for the chat. Returned only in getChat.
- `business_location` (*telebot.types.BusinessLocation*) – Optional. Business location for the chat. Returned only in getChat.

:param business_opening_hours : Optional. Business opening hours for the chat. Returned only in getChat. :type business_opening_hours: `telebot.types.BusinessHours`

Параметры

- `personal_chat` (*telebot.types.Chat*) – Optional. For private chats, the personal channel of the user. Returned only in getChat.
- `available_reactions` (`list` of *telebot.types.ReactionType*) – Optional. List of available chat reactions; for private chats, supergroups and channels. Returned only in getChat.
- `accent_color_id` (`int`) – Optional. Optional. Identifier of the accent color for the chat name and backgrounds of the chat photo, reply header, and link preview. See accent colors for more details. Returned only in getChat. Always returned in getChat.
- `background_custom_emoji_id` (`str`) – Optional. Custom emoji identifier of emoji chosen by the chat for the reply header and link preview background. Returned only in getChat.
- `profile_accent_color_id` (`int`) – Optional. Identifier of the accent color for the chat's profile background. See profile accent colors for more details. Returned only in getChat.
- `profile_background_custom_emoji_id` (`str`) – Optional. Custom emoji identifier of the emoji chosen by the chat for its profile background. 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.
- `emoji_status_expiration_date` (`int`) – Optional. Expiration date of the emoji status of the other party in a private chat, if any. 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.
- `unrestrict_boost_count` (`int`) – Optional. For supergroups, the minimum number of boosts that a non-administrator user needs to add in order to ignore slow mode and chat permissions. 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_aggressive_anti_spam_enabled` (`bool`) – Optional. `bool`, if the chat has enabled aggressive anti-spam protection. Returned only in getChat.
- `has_hidden_members` (`bool`) – Optional. `bool`, if the chat has enabled hidden members. 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.
- `has_visible_history` (`bool`) – Optional. True, if new chat members will have access to old messages; available only to chat administrators. 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.
- `custom_emoji_sticker_set_name` – Optional. For supergroups, the name of the group's custom emoji sticker set. Custom emoji from this set can be used by all users and bots in the group. Returned only in getChat.

- custom_emoji_sticker_set_name – str
- 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: bool = None*, *can_edit_messages: bool = None*, *can_pin_messages: bool = None*, *can_manage_topics: bool = None*, *can_post_stories: bool = None*, *can_edit_stories: bool = None*, *can_delete_stories: bool = None*, ***kwargs*)

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
- `can_post_stories` (`bool`) – Optional. True, if the administrator can post channel stories
- `can_edit_stories` (`bool`) – Optional. True, if the administrator can edit stories
- `can_delete_stories` (`bool`) – Optional. True, if the administrator can delete stories of other users

Результат
Instance of the class

Тип результата
telebot.types.ChatAdministratorRights

`class telebot.types.ChatBoost(`*boost_id*, *add_date*, *expiration_date*, *source*, ***kwargs*`)`

Базовые классы: *JsonDeserializable*

This object contains information about a chat boost.

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

Параметры

- `boost_id` (`str`) – Unique identifier of the boost
- `add_date` (`int`) – Point in time (Unix timestamp) when the chat was boosted
- `expiration_date` (`int`) – Point in time (Unix timestamp) when the boost will automatically expire, unless the booster's Telegram Premium subscription is prolonged
- `source` (*ChatBoostSource*) – Optional. Source of the added boost (made Optional for now due to API error)

Результат
Instance of the class

Тип результата
ChatBoost

`class telebot.types.ChatBoostAdded(`*boost_count*, ***kwargs*`)`

Базовые классы: *JsonDeserializable*

This object represents a service message about a user boosting a chat.

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

Параметры
`boost_count` (`int`) – Number of boosts added by the user

Результат
Instance of the class

Тип результата
ChatBoostAdded

`class telebot.types.ChatBoostRemoved(chat, boost_id, remove_date, source, **kwargs)`

Базовые классы: *JsonDeserializable*

This object represents a boost removed from a chat.

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

Параметры

- `chat` (*Chat*) – Chat which was boosted
- `boost_id` (`str`) – Unique identifier of the boost
- `remove_date` (`int`) – Point in time (Unix timestamp) when the boost was removed
- `source` (*ChatBoostSource*) – Source of the removed boost

Результат
Instance of the class

Тип результата
ChatBoostRemoved

`class telebot.types.ChatBoostSource`

Базовые классы: `ABC`, *JsonDeserializable*

This object describes the source of a chat boost. It can be one of
ChatBoostSourcePremium ChatBoostSourceGiftCode ChatBoostSourceGiveaway

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

Результат
Instance of the class

Тип результата
ChatBoostSourcePremium or *ChatBoostSourceGiftCode* or *ChatBoostSourceGiveaway*

`class telebot.types.ChatBoostSourceGiftCode(source, user, **kwargs)`

Базовые классы: *ChatBoostSource*

The boost was obtained by the creation of Telegram Premium gift codes to boost a chat.

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

Параметры

- `source` (`str`) – Source of the boost, always "gift_code"
- `user` (*User*) – User for which the gift code was created

Результат
Instance of the class

Тип результата
ChatBoostSourceGiftCode

`class telebot.types.ChatBoostSourceGiveaway(source, giveaway_message_id, user=None, is_unclaimed=None, **kwargs)`

Базовые классы: *ChatBoostSource*

The boost was obtained by the creation of a Telegram Premium giveaway.

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

Параметры

- `source` (`str`) – Source of the boost, always "giveaway"
- `giveaway_message_id` (`int`) – Identifier of a message in the chat with the giveaway; the message could have been deleted already. May be 0 if the message isn't sent yet.
- `user` (*User*) – User that won the prize in the giveaway if any
- `is_unclaimed` (`bool`) – True, if the giveaway was completed, but there was no user to win the prize

Результат
Instance of the class

Тип результата
ChatBoostSourceGiveaway

`class telebot.types.ChatBoostSourcePremium(source, user, **kwargs)`

Базовые классы: *ChatBoostSource*

The boost was obtained by subscribing to Telegram Premium or by gifting a Telegram Premium subscription to another user.

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

Параметры

- `source` (`str`) – Source of the boost, always "premium"
- `user` (*User*) – User that boosted the chat

Результат
Instance of the class

Тип результата
ChatBoostSourcePremium

`class telebot.types.ChatBoostUpdated(chat, boost, **kwargs)`

Базовые классы: *JsonDeserializable*

This object represents a boost added to a chat or changed.

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

Параметры

- `chat` (*Chat*) – Chat which was boosted
- `boost` (*ChatBoost*) – Infomation about the chat boost

Результат
Instance of the class

Тип результата
ChatBoostUpdated

`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*, *user_chat_id*, *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` (*telebot.types.User*) – User that sent the join request
- `user_chat_id` (`int`) – Optional. Identifier of a private chat with the user who sent the join request. 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. The bot can use this identifier for 24 hours to send messages until the join request is processed, assuming no other administrator contacted the user.
- `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_audios=None*, *can_send_documents=None*, *can_send_photos=None*, *can_send_videos=None*, *can_send_video_notes=None*, *can_send_voice_notes=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*, *can_post_stories=None*, *can_edit_stories=None*, *can_delete_stories=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

property can_manage_voice_chats

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_audios=None*, *can_send_documents=None*, *can_send_photos=None*, *can_send_videos=None*, *can_send_video_notes=None*, *can_send_voice_notes=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*, *can_post_stories=None*, *can_edit_stories=None*, *can_delete_stories=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
- `can_post_stories` (`bool`) – Optional. True, if the administrator can post channel stories
- `can_edit_stories` (`bool`) – Optional. True, if the administrator can edit stories
- `can_delete_stories` (`bool`) – Optional. True, if the administrator can delete stories of other users

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_audios=None, can_send_documents=None, can_send_photos=None, can_send_videos=None, can_send_video_notes=None, can_send_voice_notes=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, can_post_stories=None, can_edit_stories=None, can_delete_stories=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_audios=None*, *can_send_documents=None*, *can_send_photos=None*, *can_send_videos=None*, *can_send_video_notes=None*, *can_send_voice_notes=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*, *can_post_stories=None*, *can_edit_stories=None*, *can_delete_stories=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_audios=None*, *can_send_documents=None*, *can_send_photos=None*, *can_send_videos=None*, *can_send_video_notes=None*, *can_send_voice_notes=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*, *can_post_stories=None*, *can_edit_stories=None*, *can_delete_stories=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_audios=None,
                                    can_send_documents=None, can_send_photos=None,
                                    can_send_videos=None, can_send_video_notes=None,
                                    can_send_voice_notes=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,
                                    can_post_stories=None, can_edit_stories=None,
                                    can_delete_stories=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_audios=None*, *can_send_documents=None*, *can_send_photos=None*, *can_send_videos=None*, *can_send_video_notes=None*, *can_send_voice_notes=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*, *can_post_stories=None*, *can_edit_stories=None*, *can_delete_stories=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_audios` (`bool`) – True, if the user is allowed to send audios
- `can_send_documents` (`bool`) – True, if the user is allowed to send documents
- `can_send_photos` (`bool`) – True, if the user is allowed to send photos
- `can_send_videos` (`bool`) – True, if the user is allowed to send videos
- `can_send_video_notes` (`bool`) – True, if the user is allowed to send video notes
- `can_send_voice_notes` (`bool`) – True, if the user is allowed to send 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*, *via_chat_folder_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.
- `via_chat_folder_invite_link` (`bool`) – Optional. True, if the user joined the chat via a chat folder invite link

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_audios=None*, *can_send_documents=None*, *can_send_photos=None*, *can_send_videos=None*, *can_send_video_notes=None*, *can_send_voice_notes=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_audios` (`bool`) – Optional. True, if the user is allowed to send audios
- `can_send_documents` (`bool`) – Optional. True, if the user is allowed to send documents
- `can_send_photos` (`bool`) – Optional. True, if the user is allowed to send photos
- `can_send_videos` (`bool`) – Optional. True, if the user is allowed to send videos
- `can_send_video_notes` (`bool`) – Optional. True, if the user is allowed to send video notes
- `can_send_voice_notes` (`bool`) – Optional. True, if the user is allowed to send voice notes
- `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
- `can_add_web_page_previews` (`bool`) – Optional. True, if the user is allowed to add web page previews to their 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
- `can_send_media_messages` (`bool`) – deprecated.

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.ChatShared(*request_id: int*, *chat_id: int*, *title: str | None = None*, *photo: List[PhotoSize] | None = None*, *username: str | None = None*, ***kwargs*)

Базовые классы: *JsonDeserializable*

This object contains information about the chat whose identifier was shared with the bot using a *telebot.types.KeyboardButtonRequestChat* button.

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

Параметры

- `request_id` (`int`) – identifier of the request
- `chat_id` (`int`) – Identifier of the shared 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 64-bit integer or double-precision float type are safe for storing this identifier. The bot may not have access to the chat and could be unable to use this identifier, unless the chat is already known to the bot by some other means.
- `title` (`str`) – Optional. Title of the shared chat
- `photo` (`list` of *telebot.types.PhotoSize*) – Optional. Array of Photosize
- `username` (`str`) – Optional. Username of the shared chat

Результат
Instance of the class

Тип результата
telebot.types.ChatShared

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*, *thumbnail=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.
- `thumbnail` (*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

`property thumb`

class telebot.types.ExternalReplyInfo(*origin:* MessageOrigin, *chat:* Chat | *None* = *None*, *message_id: int* | *None* = *None*, *link_preview_options:* LinkPreviewOptions | *None* = *None*, *animation:* Animation | *None* = *None*, *audio:* Audio | *None* = *None*, *document:* Document | *None* = *None*, *photo: List[*PhotoSize*]* | *None* = *None*, *sticker:* Sticker | *None* = *None*, *story:* Story | *None* = *None*, *video:* Video | *None* = *None*, *video_note:* VideoNote | *None* = *None*, *voice:* Voice | *None* = *None*, *has_media_spoiler: bool* | *None* = *None*, *contact:* Contact | *None* = *None*, *dice:* Dice | *None* = *None*, *game:* Game | *None* = *None*, *giveaway:* Giveaway | *None* = *None*, *giveaway_winners:* GiveawayWinners | *None* = *None*, *invoice:* Invoice | *None* = *None*, *location:* Location | *None* = *None*, *poll:* Poll | *None* = *None*, *venue:* Venue | *None* = *None*, ***kwargs*)

Базовые классы: *JsonDeserializable*

This object contains information about a message that is being replied to, which may come from another chat or forum topic.

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

Параметры

- `origin` (*MessageOrigin*) – Origin of the message replied to by the given message
- `chat` (*Chat*) – Optional. Chat the original message belongs to. Available only if the chat is a supergroup or a channel.
- `message_id` (`int`) – Optional. Unique message identifier inside the original chat. Available only if the original chat is a supergroup or a channel.
- `link_preview_options` (*LinkPreviewOptions*) – Optional. Options used for link preview generation for the original message, if it is a text message
- `animation` (*Animation*) – Optional. Message is an animation, information about the animation
- `audio` (*Audio*) – Optional. Message is an audio file, information about the file
- `document` (*Document*) – Optional. Message is a general file, information about the file
- `photo` (`list` of *PhotoSize*) – Optional. Message is a photo, available sizes of the photo
- `sticker` (*Sticker*) – Optional. Message is a sticker, information about the sticker
- `story` (*Story*) – Optional. Message is a forwarded story
- `video` (*Video*) – Optional. Message is a video, information about the video
- `video_note` (*VideoNote*) – Optional. Message is a video note, information about the video message
- `voice` (*Voice*) – Optional. Message is a voice message, information about the file
- `has_media_spoiler` (`bool`) – Optional. True, if the message media is covered by a spoiler animation
- `contact` (*Contact*) – Optional. Message is a shared contact, information about the contact

- `dice` (*Dice*) – Optional. Message is a dice with random value
- `game` (*Game*) – Optional. Message is a game, information about the game. More about games »
- `giveaway` (*Giveaway*) – Optional. Message is a scheduled giveaway, information about the giveaway
- `giveaway_winners` (*GiveawayWinners*) – Optional. A giveaway with public winners was completed
- `invoice` (*Invoice*) – Optional. Message is an invoice for a payment, information about the invoice. More about payments »
- `location` (*Location*) – Optional. Message is a shared location, information about the location
- `poll` (*Poll*) – Optional. Message is a native poll, information about the poll
- `venue` (*Venue*) – Optional. Message is a venue, information about the venue

Результат
Instance of the class

Тип результата
ExternalReplyInfo

`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: bool | None = None*, *input_field_placeholder: str | None = 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 to a message in the same chat and forum topic, 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: str | None = None*, ***kwargs*`)`

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: str | None = None*, ***kwargs*`)`

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.ForumTopicEdited(name: str | None = None, icon_custom_emoji_id: str | None = None, **kwargs)`

Базовые классы: *JsonDeserializable*

This object represents a service message about an edited forum topic.

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

Параметры

- `name` (`str`) – Optional, Name of the topic(if updated)
- `icon_custom_emoji_id` (`str`) – Optional. New identifier of the custom emoji shown as the topic icon, if it was edited; an empty string if the icon was removed

`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.GeneralForumTopicHidden`

Базовые классы: *JsonDeserializable*

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

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

`class telebot.types.GeneralForumTopicUnhidden`

Базовые классы: *JsonDeserializable*

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

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

`class telebot.types.Giveaway(chats: List[Chat], winners_selection_date: int, winner_count: int, only_new_members: bool | None = None, has_public_winners: bool | None = None, prize_description: str | None = None, country_codes: List[str] | None = None, premium_subscription_month_count: int | None = None, **kwargs)`

Базовые классы: *JsonDeserializable*

This object represents a message about a scheduled giveaway.

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

Параметры

- `chats` (`list` of *Chat*) – The list of chats which the user must join to participate in the giveaway
- `winners_selection_date` (`int`) – Point in time (Unix timestamp) when winners of the giveaway will be selected
- `winner_count` (`int`) – The number of users which are supposed to be selected as winners of the giveaway
- `only_new_members` (`bool`) – Optional. True, if only users who join the chats after the giveaway started should be eligible to win
- `has_public_winners` (`bool`) – Optional. True, if the list of giveaway winners will be visible to everyone
- `prize_description` (`str`) – Optional. Description of additional giveaway prize
- `country_codes` (`list` of `str`) – Optional. A list of two-letter ISO 3166-1 alpha-2 country codes indicating the countries from which eligible users for the giveaway must come. If empty, then all users can participate in the giveaway.
- `premium_subscription_month_count` (`int`) – Optional. The number of months the Telegram Premium subscription won from the giveaway will be active for

Результат
Instance of the class

Тип результата
Giveaway

`class telebot.types.GiveawayCompleted(`*winner_count: int, unclaimed_prize_count: int | None = None, giveaway_message:* Message *| None = None, **kwargs*`)`

Базовые классы: *JsonDeserializable*

This object represents a service message about the completion of a giveaway without public winners.

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

Параметры

- `winner_count` (`int`) – Number of winners in the giveaway
- `unclaimed_prize_count` (`int`) – Optional. Number of undistributed prizes
- `giveaway_message` (*Message*) – Optional. Message with the giveaway that was completed, if it wasn't deleted

Результат
Instance of the class

Тип результата
GiveawayCompleted

`class telebot.types.GiveawayCreated(`***kwargs*`)`

Базовые классы: *JsonDeserializable*

This object represents a service message about the creation of a scheduled giveaway. Currently holds no information.

class telebot.types.GiveawayWinners(*chat: Chat, giveaway_message_id: int, winners_selection_date: int, winner_count: int, winners: List[User], additional_chat_count: int | None = None, premium_subscription_month_count: int | None = None, unclaimed_prize_count: int | None = None, only_new_members: bool | None = None, was_refunded: bool | None = None, prize_description: str | None = None, **kwargs*)

Базовые классы: *JsonDeserializable*

This object represents a message about the completion of a giveaway with public winners.

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

Параметры

- chat (*Chat*) – The chat that created the giveaway
- giveaway_message_id (int) – Identifier of the messsage with the giveaway in the chat
- winners_selection_date (int) – Point in time (Unix timestamp) when winners of the giveaway were selected
- winner_count (int) – Total number of winners in the giveaway
- winners (list of *User*) – List of up to 100 winners of the giveaway
- additional_chat_count (int) – Optional. The number of other chats the user had to join in order to be eligible for the giveaway
- premium_subscription_month_count (int) – Optional. The number of months the Telegram Premium subscription won from the giveaway will be active for
- unclaimed_prize_count (int) – Optional. Number of undistributed prizes
- only_new_members (bool) – Optional. True, if only users who had joined the chats after the giveaway started were eligible to win
- was_refunded (bool) – Optional. True, if the giveaway was canceled because the payment for it was refunded
- prize_description (str) – Optional. Description of additional giveaway prize

Результат

Instance of the class

Тип результата

GiveawayWinners

class telebot.types.InaccessibleMessage(*chat, message_id, date, **kwargs*)

Базовые классы: *JsonDeserializable*

This object describes a message that was deleted or is otherwise inaccessible to the bot.

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

Параметры

- chat (*Chat*) – Chat the message belonged to
- message_id (int) – Unique message identifier inside the chat
- date (int) – Always 0. The field can be used to differentiate regular and inaccessible messages.

Результат
Instance of the class

Тип результата
InaccessibleMessage

class telebot.types.InlineKeyboardButton(*text*, *url=None*, *callback_data=None*, *web_app=None*, *switch_inline_query=None*, *switch_inline_query_current_chat=None*, *switch_inline_query_chosen_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.
- `switch_inline_query_chosen_chat` (*telebot.types.SwitchInlineQueryChosenChat*) – Optional. If set, pressing the button will prompt the user to select one of their chats of the specified type, open that chat and insert the bot's username and the specified inline query in the input field

- `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({
    '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
```

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

Параметры

- `keyboard` (`list` of `list` of *telebot.types.InlineKeyboardButton*) – `list` of button rows, each represented by an `list` of *telebot.types.InlineKeyboardButton* objects
- `row_width` (`int`) – number of *telebot.types.InlineKeyboardButton* objects on each row

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*, *thumbnail_url=None*, *thumbnail_width=None*, *thumbnail_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
- thumbnail_url (str) – Optional. Url of the thumbnail for the result
- thumbnail_width (int) – Optional. Thumbnail width
- thumbnail_height (int) – Optional. Thumbnail height

Результат
Instance of the class

Тип результата
telebot.types.InlineQueryResultArticle

property thumb_height

property thumb_url

property thumb_width

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, thumbnail_url=None, thumbnail_width=None, thumbnail_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
- thumbnail_url (str) – Optional. Url of the thumbnail for the result
- thumbnail_width (int) – Optional. Thumbnail width
- thumbnail_height (int) – Optional. Thumbnail height

Результат
Instance of the class

Тип результата
telebot.types.InlineQueryResultContact

property thumb_height

property thumb_url

property thumb_width

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*, *thumbnail_url=None*, *thumbnail_width=None*, *thumbnail_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
- `thumbnail_url` (`str`) – Optional. URL of the thumbnail (JPEG only) for the file
- `thumbnail_width` (`int`) – Optional. Thumbnail width
- `thumbnail_height` (`int`) – Optional. Thumbnail height

Результат
Instance of the class

Тип результата
telebot.types.InlineQueryResultDocument

`property thumb_height`

`property thumb_url`

`property thumb_width`

`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*, *thumbnail_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*, *thumbnail_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
- `thumbnail_url` (`str`) – URL of the static (JPEG or GIF) or animated (MPEG4) thumbnail for the result
- `thumbnail_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`

`property thumb_mime_type`

`property thumb_url`

`class telebot.types.InlineQueryResultLocation(`*id, title, latitude, longitude, horizontal_accuracy, live_period=None, reply_markup=None, input_message_content=None, thumbnail_url=None, thumbnail_width=None, thumbnail_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
- `thumbnail_url` (`str`) – Optional. Url of the thumbnail for the result
- `thumbnail_width` (`int`) – Optional. Thumbnail width
- `thumbnail_height` (`int`) – Optional. Thumbnail height

Результат
Instance of the class

Тип результата
telebot.types.InlineQueryResultLocation

`property thumb_height`

`property thumb_url`

`property thumb_width`

`class telebot.types.InlineQueryResultMpeg4Gif(`*id, mpeg4_url, thumbnail_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, thumbnail_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
- `thumbnail_url` (`str`) – URL of the static (JPEG or GIF) or animated (MPEG4) thumbnail for the result
- `thumbnail_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`

`property thumb_mime_type`

`property thumb_url`

`class telebot.types.InlineQueryResultPhoto(`*id, photo_url, thumbnail_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
- `thumbnail_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`

`property thumb_url`

`class telebot.types.InlineQueryResultVenue(`*id, title, latitude, longitude, address, foursquare_id=None, foursquare_type=None, reply_markup=None, input_message_content=None, thumbnail_url=None, thumbnail_width=None, thumbnail_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
- `thumbnail_url` (`str`) – Optional. Url of the thumbnail for the result
- `thumbnail_width` (`int`) – Optional. Thumbnail width
- `thumbnail_height` (`int`) – Optional. Thumbnail height

Результат
Instance of the class

Тип результата
telebot.types.InlineQueryResultVenue

`property thumb_height`

`property thumb_url`

`property thumb_width`

`class telebot.types.InlineQueryResultVideo(`*id, video_url, mime_type, thumbnail_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"
- `thumbnail_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

property thumb_url

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.InlineQueryResultsButton(text: str, web_app: WebAppInfo | None = None, start_parameter: str | None = None)`

Базовые классы: *JsonSerializable*, *Dictionaryable*

This object represents a button to be shown above inline query results. You must use exactly one of the optional fields.

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

Параметры

- `text` (`str`) – Label text on the button
- `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 switch back to the inline mode using the method web_app_switch_inline_query inside the Web App.
- `start_parameter` (`str`) – Optional. Deep-linking parameter for the /start message sent to the bot when a user presses the 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.

Результат
Instance of the class

Тип результата
InlineQueryResultsButton

`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)
    )

# 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*, *thumbnail=None*, *caption=None*, *parse_mode=None*, *caption_entities=None*, *width=None*, *height=None*, *duration=None*, *has_spoiler=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

- `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 »
- `thumbnail` (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
- `has_spoiler` (`bool`) – Optional. True, if the uploaded animation is a spoiler

Результат

Instance of the class

Тип результата

`telebot.types.InputMediaAnimation`

`property thumb`

`class telebot.types.InputMediaAudio(`*media, thumbnail=None, caption=None, parse_mode=None, caption_entities=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

- `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 »

- `thumbnail` (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`

`property thumb`

`class telebot.types.InputMediaDocument(`*media, thumbnail=None, caption=None, parse_mode=None, caption_entities=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

- `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 »
- `thumbnail` (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`

`property thumb`

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

Базовые классы: *`InputMedia`*

Represents a photo to be sent.

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

Параметры

- `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
- `has_spoiler` (`bool`) – Optional. True, if the uploaded photo is a spoiler

Результат
Instance of the class

Тип результата
`telebot.types.InputMediaPhoto`

`class telebot.types.InputMediaVideo(`*media, thumbnail=None, caption=None, parse_mode=None, caption_entities=None, width=None, height=None, duration=None, supports_streaming=None, has_spoiler=None*`)`

Базовые классы: *`InputMedia`*

Represents a video to be sent.

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

Параметры

- `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 »
- `thumbnail` (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
- `has_spoiler` (`bool`) – Optional. True, if the uploaded video is a spoiler

Результат

Instance of the class

Тип результата

`telebot.types.InputMediaVideo`

`property thumb`

`class telebot.types.InputSticker(`*sticker: str | InputFile, emoji_list: List[str], format: str | None = None, mask_position: MaskPosition | None = None, keywords: List[str] | None = None*`)`

Базовые классы: *`Dictionaryable`*, *`JsonSerializable`*

This object describes a sticker to be added to a sticker set.

Параметры

- `sticker` (`str` or *`telebot.types.InputFile`*) – The added sticker. 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. Animated and video stickers can't be uploaded via HTTP URL.

- `emoji_list` (`list` of `str`) – One or more(up to 20) emoji(s) corresponding to the sticker
- `mask_position` (*telebot.types.MaskPosition*) – Optional. Position where the mask should be placed on faces. For "mask" stickers only.
- `keywords` (`list` of `str`) – Optional. List of 0-20 search keywords for the sticker with total length of up to 64 characters. For "regular" and "custom_emoji" stickers only.
- `format` (`str`) – Format of the added sticker, must be one of "static" for a .WEBP or .PNG image, "animated" for a .TGS animation, "video" for a WEBM video

Результат
Instance of the class

Тип результата
telebot.types.InputSticker

`convert_input_sticker()`

`class telebot.types.InputTextMessageContent(`*message_text, parse_mode=None, entities=None, disable_web_page_preview=None, link_preview_options=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`) – deprecated
- `link_preview_options` (*telebot.types.LinkPreviewOptions*) – Optional. Link preview generation options for the 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: bool | None = None*, *request_location: bool | None = None*, *request_poll:* KeyboardButtonPollType *| None = None*, *web_app:* WebAppInfo *| None = None*, *request_user:* KeyboardButtonRequestUser *| None = None*, *request_chat:* KeyboardButtonRequestChat *| None = None*, *request_users:* KeyboardButtonRequestUsers *| None = 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.
- `request_user` (*telebot.types.KeyboardButtonRequestUser*) – deprecated
- `request_users` (*telebot.types.KeyboardButtonRequestUsers*) – Optional. If specified, pressing the button will open a list of suitable users. Identifiers of selected users will be sent to the bot in a "users_shared" service message. Available in private chats only.
- `request_chat` (*telebot.types.KeyboardButtonRequestChat*) – Optional. If specified, pressing the button will open a list of suitable chats. Tapping on a chat will send its identifier to the bot in a "chat_shared" 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.KeyboardButtonRequestChat(*request_id: int*, *chat_is_channel: bool*, *chat_is_forum: bool | None = None*, *chat_has_username: bool | None = None*, *chat_is_created: bool | None = None*, *user_administrator_rights: ChatAdministratorRights | None = None*, *bot_administrator_rights: ChatAdministratorRights | None = None*, *bot_is_member: bool | None = None*, *request_title: str | None = None*, *request_photo: bool | None = None*, *request_username: bool | None = None*)

Базовые классы: *Dictionaryable*

This object defines the criteria used to request a suitable chat. The identifier of the selected chat will be shared with the bot when the corresponding button is pressed.

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

Параметры

- `request_id` (`int`) – Signed 32-bit identifier of the request, which will be received back in the ChatShared object. Must be unique within the message
- `chat_is_channel` (`bool`) – Pass True to request a channel chat, pass False to request a group or a supergroup chat.
- `chat_is_forum` (`bool`) – Optional. Pass True to request a forum supergroup, pass False to request a non-forum chat. If not specified, no additional restrictions are applied.
- `chat_has_username` (`bool`) – Optional. Pass True to request a supergroup or a channel with a username, pass False to request a chat without a username. If not specified, no additional restrictions are applied.
- `chat_is_created` (`bool`) – Optional. Pass True to request a chat owned by the user. Otherwise, no additional restrictions are applied.
- `user_administrator_rights` (*telebot.types.ChatAdministratorRights*) – Optional. A JSON-serialized object listing the required administrator rights of the user in the chat. The rights must be a superset of bot_administrator_rights. If not specified, no additional restrictions are applied.
- `bot_administrator_rights` (*telebot.types.ChatAdministratorRights*) – Optional. A JSON-serialized object listing the required administrator rights of the bot in the chat. The rights must be a subset of user_administrator_rights. If not specified, no additional restrictions are applied.
- `bot_is_member` (`bool`) – Optional. Pass True to request a chat where the bot is a member. Otherwise, no additional restrictions are applied.
- `request_title` (`bool`) – Optional. Request title
- `request_photo` (`bool`) – Optional. Request photo

- `request_username` (`bool`) – Optional. Request username

Результат
Instance of the class

Тип результата
telebot.types.KeyboardButtonRequestChat

`class telebot.types.KeyboardButtonRequestUser(`*request_id: int, user_is_bot: bool | None = None, user_is_premium: bool | None = None, max_quantity: int | None = None*`)`

Базовые классы: *KeyboardButtonRequestUsers*

Deprecated. Use KeyboardButtonRequestUsers instead.

`class telebot.types.KeyboardButtonRequestUsers(`*request_id: int, user_is_bot: bool | None = None, user_is_premium: bool | None = None, max_quantity: int | None = None, request_name: str | None = None, request_username: bool | None = None, request_photo: bool | None = None*`)`

Базовые классы: *Dictionaryable*

This object defines the criteria used to request a suitable user. The identifier of the selected user will be shared with the bot when the corresponding button is pressed.

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

Параметры

- `request_id` (`int`) – Signed 32-bit identifier of the request, which will be received back in the UsersShared object. Must be unique within the message
- `user_is_bot` (`bool`) – Optional. Pass True to request a bot, pass False to request a regular user. If not specified, no additional restrictions are applied.
- `user_is_premium` (`bool`) – Optional. Pass True to request a premium user, pass False to request a non-premium user. If not specified, no additional restrictions are applied.
- `max_quantity` (`int`) – Optional. The maximum number of users to be selected; 1-10. Defaults to 1.
- `request_name` (`bool`) – Optional. Request name
- `request_username` (`bool`) – Optional. Request username
- `request_photo` (`bool`) – Optional. Request photo

Результат
Instance of the class

Тип результата
telebot.types.KeyboardButtonRequestUsers

`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.LinkPreviewOptions(`*is_disabled: bool | None = None, url: str | None = None, prefer_small_media: bool | None = None, prefer_large_media: bool | None = None, show_above_text: bool | None = None, **kwargs*`)`

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

Describes the options used for link preview generation.

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

Параметры

- `is_disabled` (`bool`) – Optional. True, if the link preview is disabled
- `url` (`str`) – Optional. URL to use for the link preview. If empty, then the first URL found in the message text will be used
- `prefer_small_media` (`bool`) – Optional. True, if the media in the link preview is supposed to be shrunk; ignored if the URL isn't explicitly specified or media size change isn't supported for the preview
- `prefer_large_media` (`bool`) – Optional. True, if the media in the link preview is supposed to be enlarged; ignored if the URL isn't explicitly specified or media size change isn't supported for the preview
- `show_above_text` (`bool`) – Optional. True, if the link preview must be shown above the message text; otherwise, the link preview will be shown below the message text

Результат
Instance of the class

Тип результата
LinkPreviewOptions

`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=None, **kwargs)`

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=None, **kwargs)`

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, **kwargs)`

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.
- `sender_boost_count` (`int`) – Optional. If the sender of the message boosted the chat, the number of boosts added by the user
- `info` (*sender_business_bot*) – Optional. Information about the business bot that sent the message
- `date` (`int`) – Date the message was sent in Unix time
- `business_connection_id` (`str`) – Optional. Unique identifier of the business connection from which the message was received. If non-empty, the message belongs to a chat of the corresponding business account that is independent from any potential bot chat which might share the same identifier.
- `chat` (*telebot.types.Chat*) – Conversation the message belongs to
- `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.
- external_reply (*telebot.types.ExternalReplyInfo*) – Optional. Information about the message that is being replied to, which may come from another chat or forum topic
- quote (*telebot.types.TextQuote*) – Optional. For replies that quote part of the original message, the quoted part of the message
- reply_to_story (*telebot.types.Story*) – Optional. For replies to a story, the original story
- 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
- is_from_offline (bool) – Optional. True, if the message was sent by an implicit action, for example, as an away or a greeting business message, or as a scheduled message
- 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
- link_preview_options (*telebot.types.LinkPreviewOptions*) – Optional. Options used for link preview generation for the message, if it is a text message and link preview options were changed
- 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
- story (*telebot.types.Story*) – Optional. Message is a forwarded story
- 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
- `has_media_spoiler` (`bool`) – Optional. True, if the message media is covered by a spoiler animation
- `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* or *telebot.types.InaccessibleMessage*) – 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 »
- `users_shared` (*telebot.types.UsersShared*) – Optional. Service message: a user was shared with the bot
- `chat_shared` (*telebot.types.ChatShared*) – Optional. Service message: a chat was shared with the bot
- `connected_website` (`str`) – Optional. The domain name of the website on which the user has logged in. More about Telegram Login »
- `write_access_allowed` (*telebot.types.WriteAccessAllowed*) – Optional. Service message: the user allowed the bot added to the attachment menu to write messages
- `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.
- `boost_added` (*telebot.types.ChatBoostAdded*) – Optional. Service message: user boosted the chat
- `forum_topic_created` (*telebot.types.ForumTopicCreated*) – Optional. Service message: forum topic created
- `forum_topic_edited` (*telebot.types.ForumTopicEdited*) – Optional. Service message: forum topic edited
- `forum_topic_closed` (*telebot.types.ForumTopicClosed*) – Optional. Service message: forum topic closed
- `forum_topic_reopened` (*telebot.types.ForumTopicReopened*) – Optional. Service message: forum topic reopened
- `general_forum_topic_hidden` (*telebot.types.GeneralForumTopicHidden*) – Optional. Service message: the „General“ forum topic hidden

- `general_forum_topic_unhidden` (*telebot.types.GeneralForumTopicUnhidden*) – Optional. Service message: the „General“ forum topic unhidden
- `giveaway_created` (*telebot.types.GiveawayCreated*) – Optional. Service message: a giveaway has been created
- `giveaway` (*telebot.types.Giveaway*) – Optional. The message is a scheduled giveaway message
- `giveaway_winners` (*telebot.types.GiveawayWinners*) – Optional. Service message: giveaway winners(public winners)
- `giveaway_completed` (*telebot.types.GiveawayCompleted*) – Optional. Service message: giveaway completed, without public winners
- `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.

Forward_origin
Optional. For forwarded messages, information about the original message;

Результат
Instance of the class

Тип результата
telebot.types.Message

`property forward_date`

`property forward_from`

`property forward_from_chat`

`property forward_from_message_id`

`property forward_sender_name`

`property forward_signature`

`property html_caption`
Returns html-rendered caption.

`property html_text`
Returns html-rendered text.

property new_chat_member

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.

property user_shared

property voice_chat_ended

property voice_chat_participants_invited

property voice_chat_scheduled

property voice_chat_started

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)` → List[Dict] | None

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.MessageOrigin(type: str, date: int)`

Базовые классы: *JsonDeserializable*

This object describes the origin of a message.

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

Параметры

- `type` (`str`) – Type of the message origin
- `date` (`int`) – Date the message was sent originally in Unix time
- `sender_user` (*User*) – User that sent the message originally (for MessageOriginUser)
- `sender_user_name` (`str`) – Name of the user that sent the message originally (for MessageOriginHiddenUser)
- `sender_chat` (*Chat*) – Chat that sent the message originally (for MessageOriginChat)
- `author_signature` (`str`) – Optional. Author signature for certain cases

Результат
Instance of the class

Тип результата
MessageOrigin

class telebot.types.MessageOriginChannel(*date: int*, *chat:* Chat, *message_id: int*, *author_signature: str | None = None*)

Базовые классы: *MessageOrigin*

The message was originally sent to a channel chat.

Параметры

- chat (*Chat*) – Channel chat to which the message was originally sent
- message_id (int) – Unique message identifier inside the chat
- author_signature (str) – Optional. Signature of the original post author

class telebot.types.MessageOriginChat(*date: int*, *sender_chat:* Chat, *author_signature: str | None = None*)

Базовые классы: *MessageOrigin*

The message was originally sent on behalf of a chat to a group chat.

Параметры

- sender_chat (*Chat*) – Chat that sent the message originally
- author_signature (str) – Optional. For messages originally sent by an anonymous chat administrator, original message author signature

class telebot.types.MessageOriginHiddenUser(*date: int*, *sender_user_name: str*)

Базовые классы: *MessageOrigin*

The message was originally sent by an unknown user.

Параметры
sender_user_name (str) – Name of the user that sent the message originally

class telebot.types.MessageOriginUser(*date: int*, *sender_user:* User)

Базовые классы: *MessageOrigin*

The message was originally sent by a known user.

Параметры
sender_user (*User*) – User that sent the message originally

class telebot.types.MessageReactionCountUpdated(*chat:* Chat, *message_id: int*, *date: int*, *reactions: List[*ReactionCount*]*, ***kwargs*)

Базовые классы: *JsonDeserializable*

This object represents a service message about a change in the list of the current user's reactions to a message.

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

Параметры

- chat (*telebot.types.Chat*) – The chat containing the message
- message_id (int) – Unique message identifier inside the chat
- date (int) – Date of the change in Unix time
- reactions (list of *ReactionCount*) – List of reactions that are present on the message

Результат
Instance of the class

Тип результата
MessageReactionCountUpdated

class telebot.types.MessageReactionUpdated(*chat: Chat, message_id: int, date: int, old_reaction: List[ReactionType], new_reaction: List[ReactionType], user: User | None = None, actor_chat: Chat | None = None, **kwargs*)

Базовые классы: *JsonDeserializable*

This object represents a service message about a change in the list of the current user's reactions to a message.

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

Параметры

- `chat` (*telebot.types.Chat*) – The chat containing the message the user reacted to
- `message_id` (`int`) – Unique identifier of the message inside the chat
- `user` (*telebot.types.User*) – Optional. The user that changed the reaction, if the user isn't anonymous
- `actor_chat` (*telebot.types.Chat*) – Optional. The chat on behalf of which the reaction was changed, if the user is anonymous
- `date` (`int`) – Date of the change in Unix time
- `old_reaction` (`list` of *ReactionType*) – Previous list of reaction types that were set by the user
- `new_reaction` (`list` of *ReactionType*) – New list of reaction types that have been set by the user

Результат
Instance of the class

Тип результата
MessageReactionUpdated

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, option_ids, user=None, voter_chat=None, **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
- `voter_chat` (*telebot.types.Chat*) – Optional. The chat that changed the answer to the poll, if the voter is anonymous
- `user` (*telebot.types.User*) – Optional. 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.ReactionCount(*type: ReactionType*, *total_count: int*, ***kwargs*)

Базовые классы: *JsonDeserializable*

This object represents a reaction added to a message along with the number of times it was added.

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

Параметры

- type (*ReactionType*) – Type of the reaction
- total_count (int) – Number of times the reaction was added

Результат
Instance of the class

Тип результата
ReactionCount

class telebot.types.ReactionType(*type: str*)

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

This object represents a reaction type.

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

Параметры
type (str) – Type of the reaction

Результат
Instance of the class

Тип результата
ReactionType

class telebot.types.ReactionTypeCustomEmoji(*custom_emoji_id: str*, ***kwargs*)

Базовые классы: *ReactionType*

This object represents a custom emoji reaction type.

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

Параметры

- type (str) – Type of the reaction, must be custom_emoji
- custom_emoji_id (str) – Identifier of the custom emoji

Результат
Instance of the class

Тип результата
ReactionTypeCustomEmoji

class telebot.types.ReactionTypeEmoji(*emoji: str*, ***kwargs*)

Базовые классы: *ReactionType*

This object represents an emoji reaction type.

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

Параметры

- type (str) – Type of the reaction, must be emoji
- emoji (str) – Reaction emoji. List is available on the API doc.

Результат
Instance of the class

Тип результата
ReactionTypeEmoji

class telebot.types.ReplyKeyboardMarkup(*resize_keyboard: bool | None = None, one_time_keyboard: bool | None = None, selective: bool | None = None, row_width: int = 3, input_field_placeholder: str | None = None, is_persistent: bool | None = 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 to a message in the same chat and forum topic, 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.
- `is_persistent` – 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.ReplyParameters(`*message_id: int, chat_id: int | str | None = None, allow_sending_without_reply: bool | None = None, quote: str | None = None, quote_parse_mode: str | None = None, quote_entities: List[MessageEntity] | None = None, quote_position: int | None = None, **kwargs*`)`

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

Describes reply parameters for the message that is being sent.

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

Параметры

- `message_id` (`int`) – Identifier of the message that will be replied to in the current chat, or in the chat chat_id if it is specified
- `chat_id` (`int` or `str`) – Optional. If the message to be replied to is from a different chat, unique identifier for the chat or username of the channel (in the format @channelusername)
- `allow_sending_without_reply` (`bool`) – Optional. Pass True if the message should be sent even if the specified message to be replied to is not found; can be used only for replies in the same chat and forum topic.
- `quote` (`str`) – Optional. Quoted part of the message to be replied to; 0-1024 characters after entities parsing. The quote must be an exact substring of the message to be replied to, including bold, italic, underline, strikethrough, spoiler, and custom_emoji entities. The message will fail to send if the quote isn't found in the original message.
- `quote_parse_mode` (`str`) – Optional. Mode for parsing entities in the quote. See formatting options for more details.
- `quote_entities` (`list` of *MessageEntity*) – Optional. A JSON-serialized list of special entities that appear in the quote. It can be specified instead of quote_parse_mode.
- `quote_position` (`int`) – Optional. Position of the quote in the original message in UTF-16 code units

Результат
Instance of the class

Тип результата
ReplyParameters

`class telebot.types.SentWebAppMessage(`*inline_message_id=None, **kwargs*`)`

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.SharedUser(`*user_id*, *first_name=None*, *last_name=None*, *username=None*, *photo=None*, ***kwargs*`)`

Базовые классы: *JsonDeserializable*

This object contains information about a user that was shared with the bot using a KeyboardButtonRequestUser button.

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

Параметры

- `user_id` (`int`) – Identifier of the shared user. 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 64-bit integers or double-precision float types are safe for storing these identifiers. The bot may not have access to the user and could be unable to use this identifier, unless the user is already known to the bot by some other means.
- `first_name` (`str`) – Optional. First name of the user, if the name was requested by the bot
- `last_name` (`str`) – Optional. Last name of the user, if the name was requested by the bot
- `username` (`str`) – Optional. Username of the user, if the username was requested by the bot
- `photo` (`list` of *PhotoSize*) – Optional. Available sizes of the chat photo, if the photo was requested by the bot

Результат
Instance of the class

Тип результата
SharedUser

`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*, *thumbnail=None*, *emoji=None*, *set_name=None*, *mask_position=None*, *file_size=None*, *premium_animation=None*, *custom_emoji_id=None*, *needs_repainting=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
- `thumbnail` (*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
- `needs_repainting` (`bool`) – Optional. True, if the sticker must be repainted to a text color in messages, the color of the Telegram Premium badge in emoji status, white color on chat photos, or another appropriate color in other places
- `file_size` (`int`) – Optional. File size in bytes

Результат

Instance of the class

Тип результата

telebot.types.Sticker

property thumb

class telebot.types.StickerSet(*name*, *title*, *sticker_type*, *stickers*, *thumbnail=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"
- `stickers` (`list` of *`telebot.types.Sticker`*) – List of all set stickers
- `thumbnail` (*`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.

`property is_animated`

Deprecated since Bot API 7.2. Stickers can be mixed now.

`property is_video`

Deprecated since Bot API 7.2. Stickers can be mixed now.

`property thumb`

`class telebot.types.Story(`*chat:* Chat, *id: int*, ***kwargs*`)`

Базовые классы: *`JsonDeserializable`*

This object represents a story.

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

Параметры

- `chat` (*`telebot.types.Chat`*) – Chat that posted the story
- `id` (`int`) – Unique identifier for the story in the chat

Результат
Instance of the class

Тип результата
`Story`

`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.SwitchInlineQueryChosenChat(`*query=None, allow_user_chats=None, allow_bot_chats=None, allow_group_chats=None, allow_channel_chats=None, **kwargs*`)`

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

Represents an inline button that switches the current user to inline mode in a chosen chat, with an optional default inline query.

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

Параметры

- `query` (`str`) – Optional. The default inline query to be inserted in the input field. If left empty, only the bot's username will be inserted
- `allow_user_chats` (`bool`) – Optional. True, if private chats with users can be chosen
- `allow_bot_chats` (`bool`) – Optional. True, if private chats with bots can be chosen
- `allow_group_chats` (`bool`) – Optional. True, if group and supergroup chats can be chosen
- `allow_channel_chats` (`bool`) – Optional. True, if channel chats can be chosen

Результат
Instance of the class

Тип результата
SwitchInlineQueryChosenChat

`class telebot.types.TextQuote(`*text: str, position: int, entities: List[MessageEntity] | None = None, is_manual: bool | None = None, **kwargs*`)`

Базовые классы: *JsonDeserializable*

This object contains information about the quoted part of a message that is replied to by the given message.

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

Параметры

- `text` (`str`) – Text of the quoted part of a message that is replied to by the given message
- `entities` (`list` of *`MessageEntity`*) – Optional. Special entities that appear in the quote. Currently, only bold, italic, underline, strikethrough, spoiler, and custom_emoji entities are kept in quotes.
- `position` (`int`) – Approximate quote position in the original message in UTF-16 code units as specified by the sender
- `is_manual` (`bool`) – Optional. True, if the quote was chosen manually by the message sender. Otherwise, the quote was added automatically by the server.

Результат
Instance of the class

Тип результата
`TextQuote`

`property html_text`

Returns html-rendered text.

`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*, *message_reaction*, *message_reaction_count*, *removed_chat_boost*, *chat_boost*, *business_connection*, *business_message*, *edited_business_message*, *deleted_business_messages*, ***kwargs*`)`

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
- `message_reaction` (*`telebot.types.MessageReactionUpdated`*) – Optional. A reaction to a message was changed by a user. The bot must be an administrator in the chat and must explicitly specify «message_reaction» in the list of allowed_updates to receive these updates. The update isn't received for reactions set by bots.

- `message_reaction_count` (*telebot.types.MessageReactionCountUpdated*) – Optional. Reactions to a message with anonymous reactions were changed. The bot must be an administrator in the chat and must explicitly specify «message_reaction_count» in the list of allowed_updates to receive these updates.
- `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.
- `chat_boost` (*telebot.types.ChatBoostUpdated*) – Optional. A chat boost was added or changed. The bot must be an administrator in the chat to receive these updates.
- `removed_chat_boost` (`telebot.types.RemovedChatBoost`) – Optional. A chat boost was removed. The bot must be an administrator in the chat to receive these updates.
- `business_connection` (*telebot.types.BusinessConnection*) – Optional. The bot was connected to or disconnected from a business account, or a user edited an existing connection with the bot
- `business_message` (*telebot.types.Message*) – Optional. New non-service message from a connected business account
- `edited_business_message` (*telebot.types.Message*) – Optional. New version of a non-service message from a connected business account that is known to the bot and was edited

- `deleted_business_messages` (*telebot.types.Message*) – Optional. Service message: the chat connected to the business account was deleted

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, can_connect_to_business=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.
- `can_connect_to_business` (`bool`) – Optional. True, if the bot can be connected to a Telegram Business account to receive its messages. 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.UserChatBoosts(boosts, **kwargs)`

Базовые классы: *JsonDeserializable*

This object represents a list of boosts added to a chat by a user.

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

Параметры
`boosts` (`list` of *ChatBoost*) – The list of boosts added to the chat by the user

Результат
Instance of the class

Тип результата
UserChatBoosts

`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.UsersShared(request_id, user_ids: SharedUser, **kwargs)`

Базовые классы: *JsonDeserializable*

This object contains information about the users whose identifiers were shared with the bot using a KeyboardButtonRequestUsers button.

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

Параметры

- `request_id` (`int`) – Identifier of the request
- `user_ids` (`list` of `types.SharedUser`) – Array of `types.SharedUser` of the shared users. These numbers may have more than 32 significant bits and some programming languages may have difficulty/silent defects in interpreting them. But they have at most 52 significant bits, so 64-bit integers or double-precision float types are safe for storing these identifiers. The bot may not have access to the users and could be unable to use these identifiers unless the users are already known to the bot by some other means.

Результат
Instance of the class

Тип результата
UsersShared

property user_id

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*, *thumbnail=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
- thumbnail (*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

property thumb

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*, *thumbnail=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
- thumbnail (*telebot.types.PhotoSize*) – Optional. Video thumbnail
- file_size (int) – Optional. File size in bytes

Результат
Instance of the class

Тип результата
telebot.types.VideoNote

property thumb

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*, **kwargs)

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

`class telebot.types.WriteAccessAllowed(`*from_request: bool | None = None, web_app_name: str | None = None, from_attachment_menu: bool | None = None, **kwargs*`)`

Базовые классы: *JsonDeserializable*

This object represents a service message about a user allowing a bot to write messages after adding it to the attachment menu, launching a Web App from a link, or accepting an explicit request from a Web App sent by the method requestWriteAccess.

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

Параметры

- `from_request` (`bool`) – Optional. True, if the access was granted after the user accepted an explicit request from a Web App sent by the method requestWriteAccess
- `web_app_name` (`str`) – Optional. Name of the Web App which was launched from a link
- `from_attachment_menu` (`bool`) – Optional. True, if the access was granted when the bot was added to the attachment or side menu

1.3.4 Синхронный TeleBot

Методы класса TeleBot

`class telebot.ExceptionHandler`

Базовые классы: `object`

Класс для обработки исключений, возникающих во время поллинга.

`handle(exception)`

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

Базовые классы: `object`

Класс для (next step|reply) хендлеров.

`telebot.REPLY_MARKUP_TYPES`

telebot

тип
Module

alias of `InlineKeyboardMarkup` | `ReplyKeyboardMarkup` | `ReplyKeyboardRemove` | `ForceReply`

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

Базовые классы: `object`

Это основной класс для синхронного бота.

Позволяет добавить хендлеры для различных апдейтов.

Использование:

Список 4: Создание инстанса класса TeleBot

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

Больше примеров в папке examples/ : https://github.com/eternnoir/pyTelegramBotAPI/tree/master/examples

Примечание: Установите пакет coloredlogs для использования colorful_los=True

Параметры

- `token` (`str`) – Токен бота, нужно получить от @BotFather
- `parse_mode` (`str`, optional) – Глобальный parse_mode, по умолчанию None
- `threaded` (`bool`, optional) – Использовать несколько потоков, по умолчанию True
- `skip_pending` (`bool`, optional) – Игнорировать апдейты, полученные до запуска, по умолчанию False
- `num_threads` (`int`, optional) – Максимальное количество одновременно запущенных потоков, по умолчанию 2
- `next_step_backend` (`telebot.handler_backends.HandlerBackend`, optional) – Класс для сохранения next step хендлеров, по умолчанию None
- `reply_backend` (`telebot.handler_backends.HandlerBackend`, optional) – Класс для сохранения reply хендлеров, по умолчанию None
- `exception_handler` (*`telebot.ExceptionHandler`*, optional) – Класс для обработки исключений, по умолчанию None
- `last_update_id` (`int`, optional) – id последнего полученного апдейта, по умолчанию 0
- `suppress_middleware_excepions` (`bool`, optional) – Игнорировать исключения, вызванные Middleware, по умолчанию False
- `state_storage` (`telebot.storage.StateStorageBase`, optional) – Хранилище состояний (стейтов), по умолчанию StateMemoryStorage()
- `use_class_middlewares` (`bool`, optional) – Использовать Middlewares, по умолчанию False
- `disable_web_page_preview` (`bool`, optional) – Глобальное значение disable_web_page_preview, по умолчанию None
- `disable_notification` (`bool`, optional) – Глобальное значение disable_notification, по умолчанию None
- `protect_content` (`bool`, optional) – Глобальное значение protect_content, по умолчанию None
- `allow_sending_without_reply` (`bool`, optional) – Глобальное значение allow_sending_without_reply, по умолчанию None
- `colorful_logs` (`bool`, optional) – Использовать разноцветные логи

`add_custom_filter(`*custom_filter:* SimpleCustomFilter / AdvancedCustomFilter`)`

Создать кастомный фильтр.

Список 5: Пример проверки текста сообщения

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

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

Параметры

- `custom_filter` – Класс с методом check(message)
- `custom_filter` – Класс кастомного фильтра с ключом.

`add_data(user_id: int, chat_id: int | None = None, **kwargs)`

Добавить данные в состояние (стейт).

Параметры

- `user_id` (`int`) – id пользователя
- `chat_id` (`int`) – id чата
- `kwargs` – Данные для добавления

Результат

None

`add_sticker_to_set(user_id: int, name: str, emojis: List[str] | str, png_sticker: str | Any | None = None, tgs_sticker: str | Any | None = None, webm_sticker: str | Any | None = None, mask_position: MaskPosition | None = None, sticker: InputSticker | None = None) → bool`

Use this method to add a new sticker to a set created by the bot. The format of the added sticker must match the format of the other stickers in the set. Emoji sticker sets can have up to 200 stickers. Animated and video sticker sets can have up to 50 stickers. Static sticker sets can have up to 120 stickers. Returns True on success.

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

---

Примечание: ** _sticker, mask_position, emojis parameters are deprecated, use stickers instead

---

Параметры

- `user_id` (`int`) – id пользователя, создавшего стикерпак
- `name` (`str`) – Имя стикерпака
- `emojis` (`str`) – Один или несколько эмодзи, относящихся к стикеру
- `png_sticker` (`str` or `filelike object`) – Изображение стикера в формате PNG, весом не более 512 килобайт, размеры не должны превышать 512px, либо ширина, либо высота должны быть ровно 512px. Передайте file_id в формате str, чтобы отправить уже загруженный на сервера Telegram файл, передайте HTTP URL в формате str, чтобы Telegram скачал файл из интернета, или загрузите новый файл с помощью multipart/form-data.
- `tgs_sticker` (`str` or `filelike object`) – Анимированный стикер в формате TGS, загруженный с помощью multipart/form-data.
- `webm_sticker` (`str` or `filelike object`) – Анимированный стикер в формате WebM, загруженный с помощью multipart/form-data.
- `mask_position` (*`telebot.types.MaskPosition`*) – Позиция для размещения маски на лицах в формате JSON
- `sticker` (*`telebot.types.InputSticker`*) – A JSON-serialized object for sticker to be added to the sticker set

Результат

В случае успеха возвращается True.

Тип результата

`bool`

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

Используйте этот метод для отправки ответов на callback запросы, отправленные с помощью inline кнопок. Ответ будет показан пользователю как уведомление поверх чата или pop-up предупреждение.

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

Параметры

- `callback_query_id` (`int`) – Уникальный id запроса для ответа
- `text` (`str`) – Текст уведомления. если не задан, то уведомление не будет показано, 0-200 символов
- `show_alert` (`bool`) – Если True, вместо уведомления поверх чата будет показано pop-up предупреждение, по умолчанию False.
- `url` (`str`) – URL, который будет открыт пользовательским клиентом. Если вы создали игру и приняли условия через @BotFather, задайте URL, открывающий вашу игру - учитывайте, что это сработает только если запрос был отправлен с помощью callback_game кнопки.
- `cache_time` – Максимальная длительность хранения ответа на callback query пользовательским клиентом в секундах. Приложения Telegram поддерживают хранение ответов начиная с версии 3.14, по умолчанию 0.

Результат

В случае успеха возвращается True.

Тип результата

`bool`

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

Используйте этот метод для отправки ответов на inline query. В случае успеха возвращается True. Разрешено отправить не более 50 результатов на один запрос.

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

Параметры

- `inline_query_id` (`str`) – Уникальный id запроса для ответа
- `results` (`list` of `types.InlineQueryResult`) – Массив результатов для ответа на inline query
- `cache_time` (`int`) – Максимальная длительность хранения результатов inline query на сервере в секундах.
- `is_personal` (`bool`) – Передайте True, если результаты должны быть сохранены на сервере только для пользователя, отправившего запрос.
- `next_offset` (`str`) – Передайте смещение, которое клиент должен отправить в следующем запросе с таким же текстом, чтобы получить новые результаты.
- `switch_pm_parameter` (`str`) – Параметр для команды /start, отправляемой боту, когда пользователь нажимает кнопку переключения. 1-64 символа, разрешены только A-Z, a-z, 0-9, _ и -. Пример: Inline бот, который отправляет видео

с YouTube может попросить пользователя подключить бота к его YouTube аккаунту, чтобы поиск соответствовал предпочтениям пользователя. Чтобы это сделать, бот отправляет пользователю кнопку „Подключить YouTube аккаунт“ над результатами, или даже до их показа. Пользователь нажимает на кнопку, автоматически переходит в приватный чат с ботом и в это время передаёт стартовый параметр, по которому бот возвращает ссылку для авторизации (OAuth). Как только авторизация пройдена, бот может предложить switch_inline кнопку, чтобы пользователь мог легко вернуться в чат, где он хотел использовать возможности inline бота.

- `switch_pm_text` (`str`) – Параметр для передачи боту вместе с сообщением /start, отправленному при нажатии кнопки переключения
- `button` (*`types.InlineQueryResultsButton`*) – A JSON-serialized object describing a button to be shown above inline query results

Результат
В случае успеха возвращается True.

Тип результата
`bool`

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

Как только пользователь подтвердил детали оплаты и доставки, Bot API отправляет финальное подтверждение в виде апдейта с полем pre_checkout_query. Используйте этот метод для ответа на такие pre-checkout запросы. В случае успеха возвращается True.

Примечание: Bot API должно получить ответ в течение 10 секунд после отправки pre-checkout query.

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

Параметры

- `pre_checkout_query_id` (`int`) – Уникальный id запроса для ответа
- `ok` (`bool`) – Задайте True если всё правильно (выбранные товары доступны и т.д.) и бот готов обработать заказ. Задайте False если есть какие-то проблемы.
- `error_message` (`str`) – Обязательный в случае, когда ok - False. Сообщение об ошибке, которое может прочитать человек, объясняющее причину, по которой бот не может обработать заказ (например «Извините, кто-то только что купил последнюю из наших прекрасных черных футболок с коротким рукавом пока вы заполняли детали оплаты. Пожалуйста выберите другой цвет или фасон!»). Telegram покажет это сообщение пользователю.

Результат
В случае успеха возвращается True.

Тип результата
`bool`

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

Запрашивает ответ на вопрос о доставке.

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

Параметры

- `shipping_query_id` (`str`) – Уникальный id запроса для ответа
- `ok` (`bool`) – Задайте True если доставка по выбранному адресу возможна и False, если есть какие-то проблемы (например, доставка по выбранному адресу не осуществляется)
- `shipping_options` (`list` of `ShippingOption`) – Обязательный в случае, когда ok - True. Массив вариантов доставки в формате JSON.
- `error_message` (`str`) – Обязательный в случае, когда ok - False. Сообщение об ошибке, которое может прочитать человек, объясняющее причину, по которой невозможно завершить заказ (например «Извините, доставка по запрошенному адресу недоступна»). Telegram покажет это сообщение пользователю.

Результат

В случае успеха возвращается True.

Тип результата

`bool`

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

Используйте этот метод, чтобы задать результат взаимодействия с Web App и отправить соответствующее сообщение от лица пользователя в чат, из которого пришел запрос. В случае успеха возвращается объект SentWebAppMessage.

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

Параметры

- `web_app_query_id` (`str`) – Уникальный id запроса для ответа
- `result` (*`telebot.types.InlineQueryResultBase`*) – Объект в формате JSON, описывающий сообщение, которое нужно отправить

Результат

В случае успеха возвращается объект SentWebAppMessage.

Тип результата

`telebot.types.SentWebAppMessage`

`approve_chat_join_request(`*chat_id: str | int*, *user_id: int | str*`)` → bool

Используйте этот метод, чтобы одобрить запрос на вступление в чат. Бот должен быть администратором чата и иметь права администратора can_invite_users. Возвращает True в случае успеха.

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

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username супергруппы (в формате @supergroupusername)
- `user_id` (`int` or `str`) – Уникальный id сделавшего запрос пользователя

Результат

True в случае успеха.

Тип результата

`bool`

`ban_chat_member(chat_id: int / str, user_id: int, until_date: int / datetime / None = None, revoke_messages: bool / None = None)` → bool

Используйте этот метод, чтобы заблокировать пользователя в группе, супергруппе или канале. В случае супергрупп и каналов, пользователь не сможет вернуться в чат самостоятельно, используя ссылки с приглашением и т.д., пока не будет разблокирован. Возвращает True в случае успеха.

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

Параметры

- `chat_id` (`int` or `str`) – Уникальный id группы или username супергруппы или канала (в формате @channelusername)
- `user_id` (`int`) – Уникальный id сделавшего запрос пользователя
- `until_date` (`int` or `datetime`) – Дата, когда пользователь будет разблокирован, в формате UNIX time. Если пользователь заблокирован больше чем на 366 дней или меньше чем на 30 секунд, то он будет заблокирован до ручной разблокировки
- `revoke_messages` (`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
Возвращает True в случае успеха.

Тип результата
`bool`

`ban_chat_sender_chat(chat_id: int / str, sender_chat_id: int / str)` → bool

Используйте этот метод, чтобы заблокировать канал в супергруппе или канале. Владелец канала не сможет отправлять сообщения и участвовать в прямых эфирах от лица канала, пока канал не будет разблокирован. Бот должен быть администратором супергруппы или канала и иметь соответствующие права администратора.Возвращает True в случае успеха.

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

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `sender_chat_id` (`int` or `str`) – Уникальный id канала для блокировки

Результат
True в случае успеха.

Тип результата
`bool`

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

Handles new incoming business connection state.

Параметры

- `func` (`function`) – Функция, используемая в качестве фильтра
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`business_message_handler(commands: List[str] | None = None, regexp: str | None = None, func: Callable | None = None, content_types: List[str] | None = None, **kwargs)`

Handles New incoming message of any kind(for business accounts, see bot api 7.2 for more) - 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.

Пример:

Список 6: Usage of business_message_handler

```
bot = TeleBot('TOKEN')

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

# Handle all sent documents of type 'text/plain'.
@bot.business_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.business_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`) – Необязательный список строк - команд для обработки.
- `regexp` (`str`) – Необязательное регулярное выражение.
- `func` (`lambda`) – Необязательная lambda функция. Получает сообщение (объект Message) в качестве первого параметра. Функция должна вернуть True если хендлер должен обработать сообщение.
- `content_types` (`list` of `str`) – Обрабатываемые виды контента. Обязан быть списком. По умолчанию [„text“]
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
декорируемая функция

`callback_query_handler(func, **kwargs)`

Обрабатывает новый callback query. В качестве параметра передаёт в декорируемую функцию объект *telebot.types.CallbackQuery*.

Параметры

- `func` (`function`) – Функция, используемая в качестве фильтра
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

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

Обрабатывает новый пост любого типа в канале - текст, фото, стикер и т.д. В качестве параметра передаёт в декорируемую функцию объект *telebot.types.Message*.

Параметры

- `commands` (`list` of `str`) – Необязательный список строк - команд для обработки.
- `regexp` (`str`) – Необязательное регулярное выражение.
- `func` (`function`) – Функция, используемая в качестве фильтра
- `content_types` (`list` of `str`) – Обрабатываемые виды контента. Обязан быть списком. По умолчанию [„text“]
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

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

Handles new incoming chat boost state. it passes *telebot.types.ChatBoostUpdated* object.

Параметры

- `func` (`function`) – Функция, используемая в качестве фильтра
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

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

Обрабатывает запрос на вступление в чат. Бот должен иметь права администратора can_invite_users в чате, чтобы получать такие апдейты. В качестве параметра передаёт в декорируемую функцию объект *telebot.types.ChatJoinRequest*.

Параметры

- `func` (`function`) – Функция, используемая в качестве фильтра
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

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

Обрабатывает изменение статуса пользователя в чате. Бот должен быть администратором чата и явно указать “chat_member“ в allowed_updates, чтобы получать такие апдейты. В качестве параметра передаёт в декорируемую функцию объект *telebot.types.ChatMemberUpdated*.

Параметры

- `func` (`function`) – Функция, используемая в качестве фильтра
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`chosen_inline_handler(func, **kwargs)`

Обрабатывает результат inline query, который был выбран пользователем и отправлен собеседнику в чате. Пожалуйста ознакомьтесь с документацией по сбору фидбека для получения таких апдейтов вашим ботом. В качестве параметра передаёт в декорируемую функцию объект *telebot.types.ChosenInlineResult*.

Параметры

- `func` (`function`) – Функция, используемая в качестве фильтра
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`clear_reply_handlers(message: Message) → None`

Очищает список функций, зарегистрированных с помощью register_for_reply() и register_for_reply_by_message_id().

Параметры
`message` (*telebot.types.Message*) – Сообщение, у которого нужно очистить список reply хендлеров

Результат
None

`clear_reply_handlers_by_message_id(message_id: int) → None`

Очищает список функций, зарегистрированных с помощью register_for_reply() и register_for_reply_by_message_id().

Параметры
`message_id` (`int`) – id сообщения, у которого нужно очистить список reply хендлеров

Результат
None

`clear_step_handler(message: Message) → None`

Очищает список функций, зарегистрированных с помощью register_next_step_handler().

Параметры
`message` (*telebot.types.Message*) – Сообщение, после которого нужно обработать новое сообщение в том же чате.

Результат
None

`clear_step_handler_by_chat_id(chat_id: int | str) → None`

Очищает список функций, зарегистрированных с помощью register_next_step_handler().

Параметры
`chat_id` (`int` or `str`) – Чат, в котором мы хотим очистить список next step хендлеров

Результат
None

`close()` → bool

Используйте этот метод чтобы закрыть инстанс бота прежде чем перемещать его с одного локального сервера на другой. Вы должны удалить вебхук перед вызовом этого метода, чтобы убедиться. что бот не будет запущен повторно после перезапуска сервера. Метод будет возвращать ошибку 429 в течение 10 минут после запуска бота. Возвращает True в случае успеха.

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

Результат
`bool`

`close_forum_topic(`*chat_id: str | int*, *message_thread_id: int*`)` → bool

Используйте этот метод, чтобы закрыть открытый топик в чате супергруппы. Бот должен быть администратором чата и иметь права администратора can_manage_topics, за исключением случаев, когда бот является создателем топика. Возвращает True в случае успеха.

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

Aram chat_id
Уникальный id чата или username канала (в формате @channelusername)

Параметры
`message_thread_id` (`int`) – id топика для закрытия

Результат
В случае успеха возвращается True.

Тип результата
`bool`

`close_general_forum_topic(`*chat_id: int | str*`)` → bool

Используйте этот метод, чтобы закрыть открытый топик в чате супергруппы. Бот должен быть администратором чата и иметь права администратора can_manage_topics, за исключением случаев, когда бот является создателем топика. Возвращает True в случае успеха.

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

Параметры
`chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)

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

Используйте этот метод для копирования любых сообщений.

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

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `from_chat_id` (`int` or `str`) – Уникальный id чата, в который было отправлено исходное сообщение (или username канала в формате @channelusername)

- `message_id` (`int`) – id сообщения в чате, заданном в from_chat_id
- `caption` (`str`) – Новая подпись для медиа, 0-1024 символа после форматирования. Если не задано, используется исходная подпись
- `parse_mode` (`str`) – Режим форматирования новой подписи.
- `caption_entities` (Array of *telebot.types.MessageEntity*) – Список отформатированных частей новой подписи в формате JSON, можно использовать вместо parse_mode
- `disable_notification` (`bool`) – Отправить сообщение, при получении которого пользователи получат уведомление без звука.
- `protect_content` (`bool`) – Запретить пересылку и сохранение содержимого сообщения
- `reply_to_message_id` (`int`) – deprecated.
- `allow_sending_without_reply` (`bool`) – deprecated.
- `reply_markup` (*telebot.types.InlineKeyboardMarkup* or *telebot.types.ReplyKeyboardMarkup* or *telebot.types.ReplyKeyboardRemove* or *telebot.types.ForceReply*) – Дополнительные элементы интерфейса. Inline клавиатура, текстовая клавиатура, запрос на удаление текстовой клавиатуры или запрос на ответ от пользователя.
- `timeout` (`int`) – Таймаут запроса в секундах.
- `message_thread_id` (`int`) – id топика, в который нужно отправить сообщение
- `reply_parameters` (*telebot.types.ReplyParameters*) – Additional parameters for replies to messages

Результат

On success, the MessageId of the sent message is returned.

Тип результата

telebot.types.MessageID

`copy_messages`(*chat_id: str | int*, *from_chat_id: str | int*, *message_ids: List[int]*, *disable_notification: bool | None = None*, *message_thread_id: int | None = None*, *protect_content: bool | None = None*, *remove_caption: bool | None = None*) → List[*MessageID*]

Use this method to copy messages of any kind. If some of the specified messages can't be found or copied, they are skipped. Service messages, giveaway messages, giveaway winners messages, and invoice messages can't be copied. A quiz poll can be copied only if the value of the field correct_option_id is known to the bot. The method is analogous to the method forwardMessages, but the copied messages don't have a link to the original message. Album grouping is kept for copied messages. On success, an array of MessageId of the sent messages is returned.

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

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `from_chat_id` (`int` or `str`) – Уникальный id чата, в который было отправлено исходное сообщение (или username канала в формате @channelusername)
- `message_ids` (`list` of `int`) – Message identifiers in the chat specified in from_chat_id

- `disable_notification` (`bool`) – Отправить сообщение, при получении которого пользователи получат уведомление без звука
- `message_thread_id` (`int`) – Identifier of a message thread, in which the messages will be sent
- `protect_content` (`bool`) – Запретить пересылку и сохранение содержимого пересланного сообщения
- `remove_caption` (`bool`) – Pass True to copy the messages without their captions

Результат
On success, an array of MessageId of the sent messages is returned.

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

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

Используйте этот метод, чтобы создать дополнительную ссылку-приглашение в чат. Бот должен быть администратором чата и иметь соответствующие права администратора. Ссылка может быть аннулирована методом revokeChatInviteLink. Возвращает новую ссылку-приглашение (ChatInviteLink).

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

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `name` (`str`) – Название ссылки-приглашения; 0-32 символа
- `expire_date` (`int` or `datetime`) – Время, когда ссылка будет аннулирована в формате Unix timestamp
- `member_limit` (`int`) – Максимальное количество пользователей в чате
- `creates_join_request` (`bool`) – True, если пользователи, использующие эту ссылку должны быть одобрены администраторами чата. Нельзя использовать True вместе с member_limit

Результат
Возвращает новую ссылку-приглашение (ChatInviteLink).

Тип результата
`telebot.types.ChatInviteLink`

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

Используйте этот метод, чтобы создать топик в супергруппе. Бот должен быть администратором чата и иметь права администратора can_manage_topics. Возвращает информацию о созданном топике (ForumTopic).

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

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `name` (`str`) – Имя топика, 1-128 символов

- `icon_color` (`int`) – Цвет иконки топика в формате RGB. В текущий момент, доступны цвета 0x6FB9F0, 0xFFD67E, 0xCB86DB, 0x8EEE98, 0xFF93B2, or 0xFB6F5F
- `icon_custom_emoji_id` (`str`) – Кастомный эмодзи для использования в качестве иконки топика. Должно быть "tgs" эмодзи и быть ровно 1 символом

Результат

В случае успеха возвращается информация о созданном топике (ForumTopic).

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: int | None = None*, *suggested_tip_amounts: List[int] | None = None*, *provider_data: str | None = None*, *photo_url: str | None = None*, *photo_size: int | None = None*, *photo_width: int | None = None*, *photo_height: int | None = None*, *need_name: bool | None = None*, *need_phone_number: bool | None = None*, *need_email: bool | None = None*, *need_shipping_address: bool | None = None*, *send_phone_number_to_provider: bool | None = None*, *send_email_to_provider: bool | None = None*, *is_flexible: bool | None = None*) → str

используйте этот метод, чтобы создать ссылку-инвойс. Возвращает созданную ссылку в случае успеха (String).

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

Параметры

- `title` (`str`) – Название товара, 1-32 символа
- `description` (`str`) – Описание товара, 1-255 символов
- `payload` (`str`) – Дополнительные данные, 1-128 байт. Не будет показано пользователю, используйте во внутренних процессах.
- `provider_token` (`str`) – Токен платежной системы, полученный через @BotFather
- `currency` (`str`) – Трехбуквенный код валюты в формате ISO 4217, см. https://core.telegram.org/bots/payments#supported-currencies
- `prices` (`list` of *`types.LabeledPrice`*) – Детали цены, список компонент (например цена продукта, налог, скидка, стоимость доставки, налог на доставку, бонус и т.д.)
- `max_tip_amount` (`int`) – Максимальный размер чаевых в наименьших единицах выбранной валюты
- `suggested_tip_amounts` (`list` of `int`) – Массив предлагаемых вариантов чаевых в наименьших единицах выбранной валюты в формате JSON. Можно задать не более 4 вариантов. Варианты чаевых должны быть больше нуля, перечисленные в порядке строгого возрастания и не превышать max_tip_amount.
- `provider_data` (`str`) – Данные о инвойсе в формате JSON, которые будут переданы платежной системе. Подробное описание обязательных полей должно быть предоставлено провайдером платежной системы.
- `photo_url` (`str`) – URL изображения товара для инвойса. Может быть изображением товаров или изображением инвойса. Людям больше нравится видеть фото товара, за который они платят.

- `photo_size` (`int`) – Вес изображения в байтах
- `photo_width` (`int`) – Ширина изображения
- `photo_height` (`int`) – Высота изображения
- `need_name` (`bool`) – Передайте True, если для совершения заказа требуется полное имя пользователя
- `need_phone_number` (`bool`) – Передайте True, если для совершения заказа требуется номер телефона пользователя
- `need_email` (`bool`) – Передайте True, если для совершения заказа требуется email пользователя
- `need_shipping_address` (`bool`) – Передайте True, если для совершения заказа требуется адрес доставки
- `send_phone_number_to_provider` (`bool`) – Передайте True, если номер телефона пользователя нужно отправить платежной системе
- `send_email_to_provider` (`bool`) – Передайте True, если email пользователя нужно отправить платежной системе
- `is_flexible` (`bool`) – Передайте True, если окончательная цена зависит от способа доставки

Результат

Созданная ссылка-инвойс (String) в случае успеха.

Тип результата

`str`

`create_new_sticker_set`(*user_id: int*, *name: str*, *title: str*, *emojis: List[str] | None = None*, *png_sticker: Any | str = None*, *tgs_sticker: Any | str = None*, *webm_sticker: Any | str = None*, *contains_masks: bool | None = None*, *sticker_type: str | None = None*, *mask_position:* MaskPosition *| None = None*, *needs_repainting: bool | None = None*, *stickers: List[*InputSticker*] = None*, *sticker_format: str | None = None*) → bool

Используйте этот метод, чтобы создать новый стикерпак, владельцем которого станет пользователь. Бот будет иметь возможность редактировать созданный стикерпак. Возвращает True в случае успеха.

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

---

Примечание: Fields *_sticker are deprecated, pass a list of stickers to stickers parameter instead.

Параметры

- `user_id` (`int`) – id пользователя, создавшего стикерпак
- `name` (`str`) – Короткое имя стикерпака для использования в ссылках вида t.me/addstickers/ (например animals). Может содержать только латинские буквы, цифры и нижние подчеркивания. Должно начинаться с буквы, не может содержать подряд идущие нижние подчеркивания и должно заканчиваться на «_by_<bot_username>». <bot_username> учитывает регистр. 1-64 символа.
- `title` (`str`) – Название стикерпака, 1-64 символа
- `emojis` (`str`) – Один или несколько эмодзи, относящихся к стикеру

- `png_sticker` (`str`) – Изображение стикера в формате PNG, весом не более 512 килобайт, размеры не должны превышать 512px, либо ширина, либо высота должны быть ровно 512px. Передайте file_id в формате str, чтобы отправить уже загруженный на сервера Telegram файл, передайте HTTP URL в формате str, чтобы Telegram скачал файл из интернета, или загрузите новый файл с помощью multipart/form-data.
- `tgs_sticker` (`str`) – Анимированный стикер в формате TGS, загруженный с помощью multipart/form-data.
- `webm_sticker` (`str`) – Анимированный стикер в формате WebM, загруженный с помощью multipart/form-data.
- `contains_masks` (`bool`) – Передайте True, если создаётся стикерпак масок. Устарело, начиная с Bot API 6.2, используйте sticker_type.
- `sticker_type` (`str`) – Type of stickers in the set, pass "regular", "mask", or "custom_emoji". By default, a regular sticker set is created.
- `mask_position` (*`telebot.types.MaskPosition`*) – Позиция для размещения маски на лицах в формате JSON
- `needs_repainting` (`bool`) – Pass True if stickers in the sticker set must be repainted to the color of text when used in messages, the accent color if used as emoji status, white on chat photos, or another appropriate color based on context; for custom emoji sticker sets only
- `stickers` (`list` of *`telebot.types.InputSticker`*) – List of stickers to be added to the set
- `sticker_format` (`str`) – deprecated

Результат
В случае успеха возвращается True.

Тип результата
`bool`

`decline_chat_join_request(`*chat_id: str | int*, *user_id: int | str*`)` → bool

Используйте этот метод, чтобы отклонить запрос на вступление в чат. Бот должен быть администратором чата и иметь права администратора can_invite_users. Возвращает True в случае успеха.

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

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username супергруппы (в формате @supergroupusername)
- `user_id` (`int` or `str`) – Уникальный id сделавшего запрос пользователя

Результат
True в случае успеха.

Тип результата
`bool`

`delete_chat_photo(`*chat_id: int | str*`)` → bool

Используйте этот метод, чтобы удалить фото чата. Нельзя изменить фото в приватных чатах. Бот должен быть администратором чата и иметь соответствующие права администратора. Возвращает True в случае успеха. Примечание: В обычных группах (не супергруппах),

метод будет работать только в случаях, когда настройка 'All Members Are Admins' выключена.

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

Параметры
`chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)

Результат
True в случае успеха.

Тип результата
`bool`

`delete_chat_sticker_set`(*chat_id: int | str*) → bool

Используйте этот метод, чтобы удалить стикерпак группы из супергруппы. Бот должен быть администратором чата и иметь соответствующие права администратора. Используйте поле can_set_sticker_set, возвращаемое методом getChat, чтобы проверить, что бот может использовать этот метод. Возвращает True в случае успеха.

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

Параметры
`chat_id` (`int` or `str`) – Уникальный id чата или username супергруппы (в формате @supergroupusername)

Результат
Возвращает True в случае успеха.

Тип результата
`bool`

`delete_forum_topic`(*chat_id: str | int*, *message_thread_id: int*) → bool

Используйте этот метод, чтобы удалить топик в супергруппе. Бот должен быть администратором чата и иметь права администратора can_manage_topics, за исключением случае, когда бот является создателем топика. Возвращает True в случае успеха.

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

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `message_thread_id` (`int`) – id топика, который нужно удалить

Результат
В случае успеха возвращается True.

Тип результата
`bool`

`delete_message`(*chat_id: int | str*, *message_id: int*, *timeout: int | None = None*) → bool

Используйте этот метод, чтобы удалить сообщение, в том числе сервисное, ограничения: - Сообщение может быть удалено только если оно было отправлено менее 48 часов назад. - Dice-сообщение в приватном чате может быть удалено только если оно было отправлено более 24 часов назад. - Боты могут удалять свои сообщения в приватных чатах, группах и супергруппах. - Боты могут удалять чужие сообщение в приватных чатах. - Боты с правами администратора can_post_messages могут удалять сообщения в каналах. - Если бот является администратором группы, он может удалить любое сообщение в ней. - Если бот имеет права

администратора can_delete_messages в супергруппе или канале, он может удалить любое сообщение в них. Возвращает True в случае успеха.

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

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `message_id` (`int`) – id сообщения, которое нужно удалить
- `timeout` (`int`) – Таймаут запроса в секундах.

Результат
Возвращает True в случае успеха.

Тип результата
`bool`

`delete_messages(`*chat_id: int | str*, *message_ids: List[int]*`)`

Use this method to delete multiple messages simultaneously. If some of the specified messages can't be found, they are skipped. Returns True on success.

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

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `message_ids` (`list` of `int`) – Identifiers of the messages to be deleted

Результат
Возвращает True в случае успеха.

`delete_my_commands(`*scope:* BotCommandScope *| None = None*, *language_code: str | None = None*`)` → bool

Используйте этот метод, чтобы удалить список команд бота для заданных поля видимости и языка. После удаления, команды более широкого поля видимости будут доступны пользователям, которых коснулись изменения. Возвращает True в случае успеха.

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

Параметры

- `scope` (*`telebot.types.BotCommandScope`*) – Область видимости команд. По умолчанию BotCommandScopeDefault.
- `language_code` (`str`) – Двухбуквенный языковой код в формате ISO 639-1. Если не задан, изменения коснутся команд для всех пользователей в заданном поле видимости, не имеющих команд на их языке

Результат
True в случае успеха.

Тип результата
`bool`

`delete_state(`*user_id: int*, *chat_id: int | None = None*`)` → None

Удалить текущее состояние (стейт) пользователя.

Параметры

- `user_id` (`int`) – id пользователя

- `chat_id` (`int`) – id чата

Результат
None

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

Используйте этот метод, чтобы удалить стикер из стикерпака, созданного ботом. Возвращает True в случае успеха.

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

Параметры
`sticker` – id файла стикера

Результат
В случае успеха возвращается True.

Тип результата
`bool`

`delete_sticker_set`(*name: str*) → bool

Use this method to delete a sticker set. Returns True on success.

Параметры
`name` (`str`) – Имя стикерпака

Результат
Возвращает True в случае успеха.

Тип результата
`bool`

`delete_webhook`(*drop_pending_updates: bool | None = None*, *timeout: int | None = None*) → bool

Используйте этот метод, чтобы удалить вебхук, если вы решите перейти обратно на getUpdates. Возвращает True в случае успеха.

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

Параметры

- `drop_pending_updates` – Передайте True, чтобы удалить все предшествующие запуску бота апдейты, по умолчанию None
- `timeout` (`int`, optional) – Тайм-аут запроса, по умолчанию None

Результат
Возвращает True в случае успеха.

Тип результата
`bool`

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

Handles new incoming deleted messages state.

Параметры

- `func` (`function`) – Функция, используемая в качестве фильтра
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`disable_save_next_step_handlers()`

Отключить сохранение next step хендлеров (по умолчанию сохранение отключено)

Эта функция оставлена для обратной совместимости, для отключения возможности сохранения хендлеров в файл. В тех же целях MemoryHandlerBackend переопределен как новый next_step_backend вместо FileHandlerBackend.

`disable_save_reply_handlers()`

Отключить сохранение next step хендлеров (по умолчанию сохранение отключено)

Эта функция оставлена для обратной совместимости, для отключения возможности сохранения хендлеров в файл. В тех же целях MemoryHandlerBackend переопределен как новый reply_backend вместо FileHandlerBackend.

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

Скачивает файл.

Параметры
`file_path` (*str*) – Путь, куда файл нужно сохранить.

Результат
bytes

Тип результата
`bytes`

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

Используйте этот метод, чтобы изменить неосновную ссылку-приглашение, созданную ботом. Бот должен быть администратором чата и иметь соответствующие права администратора.

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

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `name` (`str`) – Название ссылки-приглашения; 0-32 символа
- `invite_link` (`str`) – Ссылка-приглашение для изменения
- `expire_date` (`int` or `datetime`) – Время, когда ссылка будет аннулирована в формате Unix timestamp
- `member_limit` (`int`) – Максимальное количество пользователей в чате
- `creates_join_request` (`bool`) – True, если пользователи, использующие эту ссылку должны быть одобрены администраторами чата. Нельзя использовать True вместе с member_limit

Результат
Возвращает новую ссылку-приглашение (ChatInviteLink).

Тип результата
telebot.types.ChatInviteLink

`edit_forum_topic(chat_id: int | str, message_thread_id: int, name: str | None = None, icon_custom_emoji_id: str | None = None)` → bool

Используйте этот метод, чтобы изменить название и иконку топика в супергруппе. Бот должен быть администратором чата и иметь права администратора can_manage_topics, кроме случаев, когда бот является создателем топика. Возвращает True в случае успеха.

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

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `message_thread_id` (`int`) – id топика для изменения
- `name` (`str`) – Необязательный, новое имя топика, 1-128 символов. Если не задано или пустое, сохранится текущее имя топика
- `icon_custom_emoji_id` (`str`) – Необязательный, новый уникальный id кастомного эмодзи, используемого в качестве иконки топика. Используйте getForumTopicIconStickers, чтобы получить все доступные id кастомных эмодзи. Передайте пустую строку, чтобы убрать иконку. Если не задан, сохранится текущая иконка топика

Результат

В случае успеха возвращается True.

Тип результата

`bool`

`edit_general_forum_topic(chat_id: int | str, name: str)` → bool

Используйте этот метод, чтобы удалить топик в супергруппе. Бот должен быть администратором чата и иметь права администратора can_manage_topics, за исключением случае, когда бот является создателем топика. Возвращает True в случае успеха.

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

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `name` (`str`) – Название товара, 1-32 символа

`edit_message_caption(caption: str, chat_id: int | str | None = None, message_id: int | None = None, inline_message_id: str | None = None, parse_mode: str | None = None, caption_entities: List[MessageEntity] | None = None, reply_markup: InlineKeyboardMarkup | None = None)` → *Message* | bool

Используйте этот метод, чтобы изменить подписи к медиа в сообщениях

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

Параметры

- `caption` (`str`) – Новая подпись к медиа
- `chat_id` (`int` | `str`) – Обязательный, если не указан inline_message_id. Уникальный id чата или username канала
- `message_id` (`int`) – Обязательный, если не указан inline_message_id.
- `inline_message_id` (`str`) – Обязательный, если не указан inline_message_id. id inline сообщения.

- `parse_mode` (`str`) – Новая подпись к медиа в сообщении, 0-1024 символа после форматирования
- `caption_entities` (`list` of *`types.MessageEntity`*) – Массив объектов, описывающих то, как будет происходить парсинг подписи к медиа в формате JSON.
- `reply_markup` (`InlineKeyboardMarkup`) – JSON-сериализованный объект inline клавиатуры.

Результат

В случае успеха если изменённое сообщение отправлено ботом, возвращается новый объект Message, иначе (inline сообщения) возвращается True.

Тип результата

`types.Message` | `bool`

`edit_message_live_location(`*latitude: float, longitude: float, chat_id: int | str | None = None, message_id: int | None = None, inline_message_id: str | None = None, reply_markup:* InlineKeyboardMarkup *| None = None, timeout: int | None = None, horizontal_accuracy: float | None = None, heading: int | None = None, proximity_alert_radius: int | None = None*`) →` *Message*

Используйте этот метод, чтобы изменить live местоположение в сообщении. Местоположение может быть изменено пока не истечёт live_period или не отключено вызовом метода stopMessageLiveLocation. В случае успеха если измененное сообщение не является inline сообщением, возвращается новый объект Message, иначе возвращается True.

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

Параметры

- `latitude` (`float`) – Широта нового местоположения
- `longitude` (`float`) – Долгота нового местоположения
- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `message_id` (`int`) – Обязательный, если не указан inline_message_id. id сообщения, которое нужно изменить
- `reply_markup` (*`telebot.types.InlineKeyboardMarkup`* or *`telebot.types.ReplyKeyboardMarkup`* or *`telebot.types.ReplyKeyboardRemove`* or *`telebot.types.ForceReply`*) – JSON-сериализованный объект новой inline клавиатуры.
- `timeout` (`int`) – Таймаут запроса в секундах.
- `inline_message_id` (`str`) – Обязательный, если не указаны chat_id и message_id. id inline сообщения
- `horizontal_accuracy` (`float`) – Радиус погрешности местоположения, измеряется в метрах; 0-1500
- `heading` (`int`) – Направление, в котором пользователь двигается, в градусах. Если указано, должно быть от 1 до 360.
- `proximity_alert_radius` (`int`) – Максимальное расстояние для показа уведомлений о приближении других участников чата, в метрах. Если указано, должно быть от 1 до 100000.

Результат
В случае успеха если измененное сообщение не является inline сообщением, возвращается новый объект Message, иначе возвращается True.

Тип результата
telebot.types.Message or bool

`edit_message_media`(*media: Any*, *chat_id: int | str | None = None*, *message_id: int | None = None*, *inline_message_id: str | None = None*, *reply_markup: InlineKeyboardMarkup | None = None*) → *Message* | bool

Используйте этот метод, чтобы изменить гифку, аудио, документ, фото или видео в сообщении. Если сообщение является частью альбома, оно может быть изменено только на фото или видео. Иначе, тип сообщения может быть изменен на любой. При изменении inline сообщения, нельзя загрузить новый файл. используйте ранее загруженные файлы через file_id или укажите URL.

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

Параметры

- `media` (`InputMedia`) – JSON-сериализованный объект нового медиа контента
- `chat_id` (`int` or `str`) – Обязательный, если не указан inline_message_id. Уникальный id чата или username канала (в формате @channelusername)
- `message_id` (`int`) – Обязательный, если не указан inline_message_id. id отправленного сообщения
- `inline_message_id` (`str`) – Обязательный, если не указаны chat_id и message_id. id inline сообщения
- `reply_markup` (*telebot.types.InlineKeyboardMarkup* or `ReplyKeyboardMarkup` or `ReplyKeyboardRemove` or `ForceReply`) – JSON-сериализованный объект inline клавиатуры.

Результат
В случае успеха если изменённое сообщение отправлено ботом, возвращается новый объект Message, иначе (inline сообщения) возвращается True.

Тип результата
types.Message or `bool`

`edit_message_reply_markup`(*chat_id: int | str | None = None*, *message_id: int | None = None*, *inline_message_id: str | None = None*, *reply_markup: InlineKeyboardMarkup | None = None*) → *Message* | bool

Используйте этот метод, чтобы изменить только reply markup сообщения.

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

Параметры

- `chat_id` (`int` or `str`) – Обязательный, если не указан inline_message_id. Уникальный id чата или username канала (в формате @channelusername)
- `message_id` (`int`) – Обязательный, если не указан inline_message_id. id отправленного сообщения
- `inline_message_id` (`str`) – Обязательный, если не указаны chat_id и message_id. id inline сообщения

- reply_markup (InlineKeyboardMarkup or ReplyKeyboardMarkup or ReplyKeyboardRemove or ForceReply) – JSON-сериализованный объект inline клавиатуры.

Результат
В случае успеха если изменённое сообщение отправлено ботом, возвращается новый объект Message, иначе (inline сообщения) возвращается True.

Тип результата
types.Message or bool

edit_message_text(*text: str*, *chat_id: int | str | None = None*, *message_id: int | None = None*, *inline_message_id: str | None = None*, *parse_mode: str | None = None*, *entities: List[MessageEntity] | None = None*, *disable_web_page_preview: bool | None = None*, *reply_markup: InlineKeyboardMarkup | None = None*, *link_preview_options: LinkPreviewOptions | None = None*) → *Message* | bool

Используйте этот метод, чтобы изменить текстовые и игровые сообщения.

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

Параметры

- text (str) – Новый текст сообщения, 1-4096 символов после форматирования
- chat_id (int or str) – Обязательный, если не указан inline_message_id. Уникальный id чата или username канала (в формате @channelusername)
- message_id (int) – Обязательный, если не указан inline_message_id. id отправленного сообщения
- inline_message_id (str) – Обязательный, если не указаны chat_id и message_id. id inline сообщения
- parse_mode (str) – Режим форматирования в тексте сообщения.
- entities (List of *telebot.types.MessageEntity*) – Список отформатированных частей в тексте сообщения, можно использовать вместо parse_mode
- disable_web_page_preview (bool) – deprecated.
- reply_markup (InlineKeyboardMarkup) – JSON-сериализованный объект inline клавиатуры.
- link_preview_options (LinkPreviewOptions) – A JSON-serialized object for options used to automatically generate previews for links.

Результат
В случае успеха если изменённое сообщение отправлено ботом, возвращается новый объект Message, иначе (inline сообщения) возвращается True.

Тип результата
types.Message or bool

edited_business_message_handler(*commands=None*, *regexp=None*, *func=None*, *content_types=None*, ***kwargs*)

Handles new version of a message(business accounts) 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) – Необязательный список строк - команд для обработки.
- regexp (str) – Необязательное регулярное выражение.

- `func` (`function`) – Функция, используемая в качестве фильтра
- `content_types` (`list` of `str`) – Обрабатываемые виды контента. Обязан быть списком. По умолчанию [„text“]
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

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

Обрабатывает новую версию поста в канале, который доступен боту и был изменён. В качестве параметра, передаёт в декорируемую функцию объект *`telebot.types.Message`*.

Параметры

- `commands` (`list` of `str`) – Необязательный список строк - команд для обработки.
- `regexp` (`str`) – Необязательное регулярное выражение.
- `func` (`function`) – Функция, используемая в качестве фильтра
- `content_types` (`list` of `str`) – Обрабатываемые виды контента. Обязан быть списком. По умолчанию [„text“]
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат

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

Обрабатывает новую версию сообщения, которое доступно боту и было изменено. В качестве параметра, передаёт в декорируемую функцию объект *`telebot.types.Message`*.

Параметры

- `commands` (`list` of `str`) – Необязательный список строк - команд для обработки.
- `regexp` (`str`) – Необязательное регулярное выражение.
- `func` (`function`) – Функция, используемая в качестве фильтра
- `content_types` (`list` of `str`) – Обрабатываемые виды контента. Обязан быть списком. По умолчанию [„text“]
- `chat_types` (`list` of `str`) – список видов чатов
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`enable_save_next_step_handlers(`*delay: int | None = 120, filename: str | None = './.handler-saves/step.save'*`)`

Разрешить сохранение next step хендлеров (по умолчанию сохранение отключено)

Эта функция, целью которой было включить возможность сохранения файлов для обработчиков, явно назначает FileHandlerBackend (вместо Saver) просто для сохранения обратной совместимости. Та же реализация теперь доступна с FileHandlerBackend.

Параметры

- `delay` (`int`, optional) – Задержка между изменениями в хендлерах и сохранении, по умолчанию 120

- `filename` (`str`, optional) – Имя файла для сохранения, по умолчанию «./.handler-saves/step.save»

Результат
None

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

Разрешить сохранение reply хендлеров (по умолчанию сохранение отключено)

Эта функция, целью которой было включить возможность сохранения файлов для обработчиков, явно назначает FileHandlerBackend (вместо Saver) просто для сохранения обратной совместимости. Та же реализация теперь доступна с FileHandlerBackend.

Параметры

- `delay` (`int`, optional) – Задержка между изменениями в хендлерах и сохранении, по умолчанию 120
- `filename` (`str`, optional) – Имя файла для сохранения, по умолчанию «./.handler-saves/reply.save»

`enable_saving_states`(*filename: str | None = './.state-save/states.pkl'*)

Разрешить сохранение стейтов (по умолчанию сохранение отключено)

Примечание: Рекомендуется передавать экземпляр класса `StatePickleStorage` в качестве state_storage при инициализации класса TeleBot вместо использования этой функции.

Параметры
`filename` (`str`, optional) – Имя файла для сохранения, по умолчанию «./.state-save/states.pkl»

`export_chat_invite_link`(*chat_id: int | str*) → str

Используйте этот метод, чтобы создать или заменить главную ссылку-приглашение в супергруппу или канал, созданную ботом. Бот должен быть администратором чата и иметь соответствующие права администратора.

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

Параметры
`chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)

Результат
новая ссылка-приглашение (String) в случае успеха.

Тип результата
`str`

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

Используйте этот метод, чтобы переслать любое сообщение.

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

Параметры

- `disable_notification` (`bool`) – Отправить сообщение, при получении которого пользователи получат уведомление без звука

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `from_chat_id` (`int` or `str`) – Уникальный id чата, в который было отправлено исходное сообщение (или username канала в формате @channelusername)
- `message_id` (`int`) – id сообщения в чате, заданном в from_chat_id
- `protect_content` (`bool`) – Запретить пересылку и сохранение содержимого пересланного сообщения
- `timeout` (`int`) – Таймаут запроса в секундах.
- `message_thread_id` (`int`) – id топика, в который нужно отправить сообщение

Результат

В случае успеха возвращает отправленное сообщение (Message).

Тип результата

telebot.types.Message

`forward_messages`(*chat_id: str | int*, *from_chat_id: str | int*, *message_ids: List[int]*, *disable_notification: bool | None = None*, *message_thread_id: int | None = None*, *protect_content: bool | None = None*) → List[*MessageID*]

Use this method to forward multiple messages of any kind. If some of the specified messages can't be found or forwarded, they are skipped. Service messages and messages with protected content can't be forwarded. Album grouping is kept for forwarded messages. On success, an array of MessageId of the sent messages is returned.

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

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `from_chat_id` (`int` or `str`) – Уникальный id чата, в который было отправлено исходное сообщение (или username канала в формате @channelusername)
- `message_ids` (`list`) – Message identifiers in the chat specified in from_chat_id
- `disable_notification` (`bool`) – Отправить сообщение, при получении которого пользователи получат уведомление без звука
- `message_thread_id` (`int`) – Identifier of a message thread, in which the messages will be sent
- `protect_content` (`bool`) – Запретить пересылку и сохранение содержимого пересланного сообщения

Результат

В случае успеха возвращает отправленное сообщение (Message).

Тип результата

telebot.types.MessageID

`get_business_connection`(*business_connection_id: str*) → *BusinessConnection*

Use this method to get information about the connection of the bot with a business account. Returns a BusinessConnection object on success.

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

Параметры

`business_connection_id` (`str`) – Unique identifier of the business connection

Результат
Returns a BusinessConnection object on success.

Тип результата
telebot.types.BusinessConnection

`get_chat(chat_id: int | str) → Chat`

Используйте этот метод, чтобы получить актуальную информацию о чате (текущее имя пользователя для персональных диалогов, текущий username пользователя, группы или канала и т.д.). В случае успеха возвращает объект Chat.

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

Параметры
`chat_id` (`int` or `str`) – Уникальный id чата или username супергруппы или канала (в формате @channelusername)

Результат
Информация о чате

Тип результата
telebot.types.Chat

`get_chat_administrators(chat_id: int | str) → List[ChatMember]`

Используйте этот метод, чтобы получить список администраторов чата. В случае успеха возвращает массив объектов ChatMember, содержащих информацию обо всех администраторах чата, кроме других ботов.

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

Параметры
`chat_id` – Уникальный id чата или username супергруппы или канала (в формате @channelusername)

Результат
Список объектов ChatMember.

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

`get_chat_member(chat_id: int | str, user_id: int) → ChatMember`

Используйте этот метод, чтобы получить информацию об участнике чата. Возвращает объект ChatMember в случае успеха.

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

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username супергруппы (в формате @supergroupusername)
- `user_id` (`int`) – Уникальный id сделавшего запрос пользователя

Результат
Возвращает объект ChatMember в случае успеха.

Тип результата
telebot.types.ChatMember

`get_chat_member_count(chat_id: int | str) → int`

Используйте этот метод, чтобы получить количество участников чата.

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

Параметры
`chat_id` (`int` or `str`) – Уникальный id чата или username супергруппы или канала (в формате @channelusername)

Результат
Количество участников чата.

Тип результата
`int`

`get_chat_members_count(**kwargs)`

`get_chat_menu_button(chat_id: int | str = None)` → *MenuButton*

Используйте этот метод, чтобы получить текущее значение кнопки menu в приватном чате, или кнопку menu по умолчанию. Возвращает MenuButton в случае успеха.

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

Параметры
`chat_id` (`int` or `str`) – Уникальный id приватного чата. Если не указан, будет возвращена кнопка menu по умолчанию.

Результат
types.MenuButton

Тип результата
`telebot.types.MenuButton`

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

Используйте этот метод, чтобы получить информацию о кастомных эмодзи по их id. Возвращает массив объектов Sticker.

Параметры
`custom_emoji_ids` (`list` of `str`) – Список id кастомных эмодзи. Можно указать не более 200 id.

Результат
Возвращает массив объектов Sticker.

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

`get_file(file_id: str | None)` → *File*

Используйте этот метод, чтобы получить базовую информацию о файле и подготовить его к скачиванию. На текущий момент, боты могут скачивать файлы весом до 20MB. В случае успеха возвращается объект File. Гарантируется, что ссылка на скачивание будет актуальна как минимум 1 час. Когда ссылка перестаёт быть актуальной, новая может быть снова запрошена с помощью get_file.

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

Параметры
`file_id` (`str`) – id файла

Результат
`telebot.types.File`

`get_file_url(file_id: str | None)` → str

Получить актуальную ссылку для скачивания файла.

Параметры
`file_id` (`str`) – id файла для получения ссылки на скачивание.

Результат
Ссылка для скачивания файла.

Тип результата
`str`

`get_forum_topic_icon_stickers()` → List[*Sticker*]

Используйте этот метод, чтобы получить кастомные эмодзи, которые могут быть использованы любыми пользователями в качестве иконок топиков. Не требует параметров. Возвращает массив объектов Sticker.

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

Результат
В случае успеха возвращается список объектов StickerSet.

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

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

Используйте этот метод, чтобы получить данные для таблицы рекордов. Вернёт очки указанного пользователя и несколько соседних результатов. В случае успеха возвращает массив объектов GameHighScore.

На текущий момент этот метод вернёт очки указанного пользователя и по два соседних результата с каждой стороны. Также вернет результаты трёх лучших игроков, если результат пользователя и соседние не являются тремя лучшими. Пожалуйста учитывайте, что это поведение может быть изменено.

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

Параметры

- `user_id` (`int`) – id пользователя
- `chat_id` (`int` or `str`) – Обязательный, если не указан inline_message_id. Уникальный id чата или username канала (в формате @channelusername)
- `message_id` (`int`) – Обязательный, если не указан inline_message_id. id отправленного сообщения
- `inline_message_id` (`str`) – Обязательный, если не указаны chat_id и message_id. id inline сообщения

Результат
В случае успеха возвращает массив объектов GameHighScore.

Тип результата
List[*types.GameHighScore*]

`get_me()` → *User*

Простой метод для тестирования токена бота. Не требует параметров. Возвращает базовую информацию о боте в виде объекта User.

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

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

Используйте этот метод, чтобы получить текущий список команд бота. Возвращает список объектов BotCommand в случае успеха.

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

Параметры

- `scope` (*telebot.types.BotCommandScope*) – Область видимости команд. По умолчанию BotCommandScopeDefault.
- `language_code` (`str`) – Двухбуквенный языковой код в формате ISO 639-1. Если не задан, изменения коснутся команд для всех пользователей в заданном поле видимости, не имеющих команд на их языке

Результат
Список объектов BotCommand в случае успеха.

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

`get_my_default_administrator_rights`(*for_channels: bool | None = None*) → *ChatAdministratorRights*

Используйте этот метод, чтобы получить текущие права администратора для бота по умолчанию. Возвращает объект ChatAdministratorRights в случае успеха.

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

Параметры
`for_channels` (`bool`) – Передайте True, чтобы получить права администратора для бота по умолчанию в каналах. Иначе, будут возвращены права администратора для бота по умолчанию в группах и супергруппах.

Результат
Возвращает объект ChatAdministratorRights в случае успеха.

Тип результата
telebot.types.ChatAdministratorRights

`get_my_description`(*language_code: str | None = None*)

Use this method to get the current bot description for the given user language. Returns BotDescription on success.

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

Параметры
`language_code` (`str`) – A two-letter ISO 639-1 language code or an empty string

Результат
telebot.types.BotDescription

`get_my_name`(*language_code: str | None = None*)

Use this method to get the current bot name for the given user language. Returns BotName on success.

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

Параметры
`language_code` (`str`) – Optional. A two-letter ISO 639-1 language code or an empty string

Результат
telebot.types.BotName

`get_my_short_description(language_code: str | None = None)`

Use this method to get the current bot short description for the given user language. Returns BotShortDescription on success.

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

Параметры

`language_code` (`str`) – A two-letter ISO 639-1 language code or an empty string

Результат

telebot.types.BotShortDescription

`get_state(user_id: int, chat_id: int | None = None)` → int | str | *State* | None

Получает текущее состояние (стейт) пользователя. Не рекомендуется использовать этот метод. Но это удобно для дебага.

Параметры

- `user_id` (`int`) – id пользователя
- `chat_id` (`int`) – id чата

Результат

состояние (стейт) пользователя

Тип результата

`int` or `str` or `telebot.types.State`

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

Используйте этот метод, чтобы получить стикерпак. В случае успеха возвращается объект StickerSet.

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

Параметры

`name` (`str`) – Имя стикерпака

Результат

В случае успеха возвращается объект StickerSet.

Тип результата

telebot.types.StickerSet

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

Используйте этот метод, чтобы получить новые апдейты с помощью long polling-а (wiki). Возвращается массив объектов Update.

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

Параметры

- `offset` (`int`, optional) – id первого апдейта. Должен быть на единицу больше наибольшего id среди ранее полученных апдейтов. По умолчанию, возвращается список апдейтов, начиная с самого раннего неполученного. Апдейт считается полученным как только вызван метод getUpdates со смещением больше, чем id этого апдейта. Отрицательное смещение может быть указано для получения последних offset апдейтов. Все предыдущие апдейты будут считаться полученными.
- `limit` (`int`, optional) – Максимальное число апдейтов для получения. Допускаются значения от 1 до 100. По умолчанию 100.

- `timeout` (`int`, optional) – Тайм-аут запроса
- `allowed_updates` (`list`, optional) – Массив строк. Список видов апдейтов, которые вы хотите получать.
- `long_polling_timeout` (`int`, optional) – Тайм-аут поллинга в секундах.

Результат
Возвращается массив объектов Update.

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

`get_user_chat_boosts(`*chat_id: int | str, user_id: int*`) → `*UserChatBoosts*

Use this method to get the list of boosts added to a chat by a user. Requires administrator rights in the chat. Returns a UserChatBoosts object.

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

Параметры

- `chat_id` (`int | str`) – Уникальный id чата или username канала
- `user_id` (`int`) – Уникальный id сделавшего запрос пользователя

Результат
On success, a UserChatBoosts object is returned.

Тип результата
`telebot.types.UserChatBoosts`

`get_user_profile_photos(`*user_id: int, offset: int | None = None, limit: int | None = None*`) → `*UserProfilePhotos*

Используйте этот метод, чтобы получить список аватарок пользователя. Возвращает объект *`telebot.types.UserProfilePhotos`*.

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

Параметры

- `user_id` (`int`) – Уникальный id сделавшего запрос пользователя
- `offset` (`int`) – Порядковый номер первого фото для получения. По умолчанию, возвращаются все фото.
- `limit` (`int`) – Максимальное число фото для получения. Допускаются значения от 1 до 100. По умолчанию 100.

Результат
UserProfilePhotos

Тип результата
`telebot.types.UserProfilePhotos`

`get_webhook_info(`*timeout: int | None = None*`) → `*WebhookInfo*

Используйте этот метод, чтобы получить текущий статус вебхука. Не требует параметров. В случае успеха возвращает объект WebhookInfo. Если бот использует getUpdates, вернёт объект с пустым атрибутом url.

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

Параметры
`timeout` (`int`, optional) – Тайм-аут запроса

Результат
В случае успеха возвращает объект WebhookInfo.

Тип результата
telebot.types.WebhookInfo

`hide_general_forum_topic(chat_id: int | str) → bool`

Используйте этот метод, чтобы удалить топик в супергруппе. Бот должен быть администратором чата и иметь права администратора can_manage_topics, за исключением случае, когда бот является создателем топика. Возвращает True в случае успеха.

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

Параметры
`chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)

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

Запустить поллинг в бесконечном цикле с обработкой исключений, чтобы избежать непредвиденных остановок поллинга.

Примечание: Установите watchdog и psutil, чтобы использовать restart_on_change.

Параметры

- `timeout` (`int`) – Тайм-аут запроса.
- `long_polling_timeout` (`int`) – Тайм-аут поллинга в секундах (см. документацию API)
- `skip_pending` (`bool`) – пропускать старые апдейты
- `logger_level` (`int.`) – Кастомный (отличающийся от логгера) уровень логирования для infinity_polling. Используйте уровни из logging в качестве значений. None/NOTSET = не логировать ошибки.
- `allowed_updates` (`list of str`) – Список видов апдейтов, которые вы хотите получать. Например, укажите ["message", "edited_channel_post", "callback_query"], чтобы получать апдейты только этих видов. Полный список доступных видов апдейтов - util.update_types. Укажите пустой список, чтобы получать все апдейты, кроме chat_member (по умолчанию). Если не задан, будет использована последняя настройка. Пожалуйста учитывайте, что этот параметр не влияет на апдейты, отправленные до вызова get_updates, поэтому нежелательные апдейты могут быть получены в течение короткого периода времени.
- `restart_on_change` (`bool`) – Перезапуск при изменении файлов. По умолчанию False
- `path_to_watch` (`str`) – Путь для мониторинга изменений. По умолчанию текущая директория.

Результат

`inline_handler(func, **kwargs)`

Обрабатывает inline query. В качестве параметра, передаёт в декорируемую функцию объект *telebot.types.InlineQuery*.

Параметры

- `func` (`function`) – Функция, используемая в качестве фильтра
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`kick_chat_member(**kwargs)`

`leave_chat(chat_id: int | str) → bool`

Используйте этот метод, чтобы покинуть группу, супергруппу или канал. Возвращает True в случае успеха.

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

Параметры
`chat_id` (`int` or `str`) – Уникальный id чата или username супергруппы или канала (в формате @channelusername)

Результат
`bool`

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

Загрузить next step хендлеры из файла

Эта функция оставлена для обратной совместимости, для загрузки хендлеров из файла с помощью FileHandlerBackend и рекомендуется к использованию только если next_step_backend был определён как FileHandlerBackend до вызова этой функции

Параметры

- `filename` (`str`, optional) – Имя файла, в котором были сохранены хендлеры, по умолчанию «./.handler-saves/step.save»
- `del_file_after_loading` (`bool`, optional) – Если передано True, файл будет удалён после загрузки, по умолчанию True

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

Загрузить reply хендлеры из файла

Эта функция оставлена для обратной совместимости, для загрузки хендлеров из файла с помощью FileHandlerBackend и рекомендуется к использованию только если reply_backend был определён как FileHandlerBackend до вызова этой функции

Параметры

- `filename` (`str`, optional) – Имя файла, в котором были сохранены хендлеры, по умолчанию «./.handler-saves/reply.save»
- `del_file_after_loading` (`bool`, optional) – Если передано True, файл будет удалён после загрузки, по умолчанию True

`log_out() → bool`

Используйте этот метод, чтобы отключиться от облачного Bot API сервера перед локальным запуском бота. Вы ДОЛЖНЫ отключить бота перед тем, как запускать его локально, иначе нет никаких гарантий, что бот будет получать апдейты. После успешного вызова, вы

можете тут же подключиться к локальному серверу, но не сможете подключиться обратно к облачному Bot API серверу в течение 10 минут. Возвращает True в случае успеха.

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

Результат
True в случае успеха.

Тип результата
`bool`

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

Обрабатывает входящие сообщения всех видов - text, photo, sticker, и т.д. В качестве параметра передаёт в декорируемую функцию объект *telebot.types.Message*. Все хендлеры сообщений проверяются в том порядке, в котором были добавлены.

Пример:

Список 7: Использование 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`) – Необязательный список строк - команд для обработки.
- `regexp` (`str`) – Необязательное регулярное выражение.
- `func` (`lambda`) – Необязательная lambda функция. Получает сообщение (объект Message) в качестве первого параметра. Функция должна вернуть True если хендлер должен обработать сообщение.

- content_types (list of str) – Обрабатываемые виды контента. Обязан быть списком. По умолчанию [„text“]
- chat_types (list of str) – список видов чатов
- kwargs – Необязательные именованные аргументы(кастомные фильтры)

Результат
декорируемая функция

message_reaction_count_handler(*func=None*, ***kwargs*)

Handles new incoming message reaction count. As a parameter to the decorator function, it passes *telebot.types.MessageReactionCountUpdated* object.

Параметры

- func (function) – Функция, используемая в качестве фильтра
- kwargs – Необязательные именованные аргументы(кастомные фильтры)

Результат

message_reaction_handler(*func=None*, ***kwargs*)

Handles new incoming message reaction. As a parameter to the decorator function, it passes *telebot.types.MessageReactionUpdated* object.

Параметры

- func (function) – Функция, используемая в качестве фильтра
- kwargs – Необязательные именованные аргументы(кастомные фильтры)

Результат

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

Функция-декоратор для middleware хендлера.

Этот декоратор может быть использован, чтобы декорировать функции, которые будут использоваться в качестве middleware перед обработкой апдейтов, будьте аккуратны и проверяйте вид апдейта внутри функции если возможны апдейты разных видов

Пример:

Список 8: Использование 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) – Необязательный список видов апдейтов, которые будут обработаны этим middleware хендлером.

Результат
функция

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

Обрабатывает изменения статуса бота. Для приватных чатов, этот апдейт отправляется только когда бот был заблокирован или разблокирован пользователем. В качестве параметра передаёт в декорируемую функцию объект *telebot.types.ChatMemberUpdated*.

Параметры

- `func` (`function`) – Функция, используемая в качестве фильтра
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`pin_chat_message(chat_id: int | str, message_id: int, disable_notification: bool | None = False) → bool`

Используйте этот метод, чтобы закрепить сообщение в супергруппе. Бот должен быть администратором чата и иметь соответствующие права администратора. Возвращает True в случае успеха.

Документация Telegram: https://core.telegram.org/bots/api#pinchatmessage

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `message_id` (`int`) – id сообщения, которое нужно закрепить
- `disable_notification` (`bool`) – Передайте True, если всем участникам группы необходимо отправить уведомление о закреплённом сообщении

Результат
True в случае успеха.

Тип результата
`bool`

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

Обрабатывает изменения ответа пользователя в не анонимном опросе(когда пользователь меняет выбор). Боты получают новые ответы только в опросах, которые отправили сами. В качестве параметра передаёт в декорируемую функцию объект *telebot.types.PollAnswer*.

Параметры

- `func` (`function`) – Функция, используемая в качестве фильтра
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`poll_handler(func, **kwargs)`

Обрабатывает изменения в состоянии опроса. Боты получают только апдейты о завершенных опросах и опросах, которые отправили сами. В качестве параметра передаёт в декорируемую функцию объект *telebot.types.Poll*.

Параметры

- `func` (`function`) – Функция, используемая в качестве фильтра

- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`polling`(*non_stop: bool | None = False*, *skip_pending: bool | None = False*, *interval: int | None = 0*, *timeout: int | None = 20*, *long_polling_timeout: int | None = 20*, *logger_level: int | None = 40*, *allowed_updates: List[str] | None = None*, *none_stop: bool | None = None*, *restart_on_change: bool | None = False*, *path_to_watch: str | None = None*)

Эта функция создаёт новый Thread, который вызывает служебную функцию __retrieve_updates. Это позволяет боту получать апдейты (Update) автоматически и вызывать соответствующие листенеры и хендлеры.

Предупреждение: Не вызывайте эту функцию более одного раза!

Всегда получает апдейты.

Не рекомендуется, начиная с версии 4.1.1: Используйте *infinity_polling()*.

Примечание: Установите watchdog и psutil, чтобы использовать restart_on_change.

Параметры

- `interval` (`int`) – Задержка между получением апдейтов
- `non_stop` (`bool`) – Не останавливать поллинг при возникновении ApiException.
- `timeout` (`int`) – Тайм-аут запроса
- `skip_pending` (`bool`) – пропускать старые апдейты
- `long_polling_timeout` (`int`) – Тайм-аут поллинга в секундах (см. документацию API)
- `logger_level` (`int`) – Кастомный (отличающийся от логгера) уровень логирования для infinity_polling. Используйте уровни из logging в качестве значений. None/NOTSET = не логировать ошибки.
- `allowed_updates` (`list of str`) – Список видов апдейтов, которые вы хотите получать. Например, укажите ["message", "edited_channel_post", "callback_query"], чтобы получать апдейты только этих видов. Полный список доступных видов апдейтов - util.update_types. Укажите пустой список, чтобы получать все апдейты, кроме chat_member (по умолчанию). Если не задан, будет использована последняя настройка. Пожалуйста учитывайте, что этот параметр не влияет на апдейты, отправленные до вызова get_updates, поэтому нежелательные апдейты могут быть получены в течение короткого периода времени.
- `none_stop` (`bool`) – deprecated.
- `restart_on_change` (`bool`) – Перезапуск при изменении файлов. По умолчанию False
- `path_to_watch` (`str`) – Путь для мониторинга изменений. По умолчанию None

Результат

`pre_checkout_query_handler`(*func*, ***kwargs*)

Новая pre-checkout query. Содержит полную информацию о заказе. В качестве параметра передаёт в декорируемую функцию объект *telebot.types.PreCheckoutQuery*.

Параметры

- `func` (`function`) – Функция, используемая в качестве фильтра
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`process_new_updates(updates: List[Update])`

Обрабатывает новые апдейты. Просто передайте список апдейтов(Update и его наследники).

Параметры
`updates` (`list` of *`telebot.types.Update`*) – Список объектов *`telebot.types.Update`*.

возвращает None

`promote_chat_member(chat_id: int | str, user_id: int, can_change_info: bool | None = None, can_post_messages: bool | None = None, can_edit_messages: bool | None = None, can_delete_messages: bool | None = None, can_invite_users: bool | None = None, can_restrict_members: bool | None = None, can_pin_messages: bool | None = None, can_promote_members: bool | None = None, is_anonymous: bool | None = None, can_manage_chat: bool | None = None, can_manage_video_chats: bool | None = None, can_manage_voice_chats: bool | None = None, can_manage_topics: bool | None = None, can_post_stories: bool | None = None, can_edit_stories: bool | None = None, can_delete_stories: bool | None = None) → bool`

Используйте этот метод, чтобы повысить или понизить пользователя в супергруппе или канале. Бот должен быть администратором чата и иметь соответствующие права администратора. Передайте False во все boolean параметры, чтобы понизить пользователя.

Документация Telegram: https://core.telegram.org/bots/api#promotechatmember

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `user_id` (`int`) – Уникальный id сделавшего запрос пользователя
- `can_change_info` (`bool`) – Передайте True, если администратор может менять название чата, аватарку и другие настройки
- `can_post_messages` (`bool`) – Передайте True, если администратор может создавать посты в канале, только для каналов
- `can_edit_messages` (`bool`) – Передайте True, если администратор может изменять сообщения других пользователей, только для каналов
- `can_delete_messages` (`bool`) – Передайте True, если администратор может удалять сообщения других пользователей
- `can_invite_users` (`bool`) – Передайте True, если администратор может приглашать новых пользователей в чат
- `can_restrict_members` (`bool`) – Передайте True, если администратор может ограничивать, банить или разбанивать участников чата
- `can_pin_messages` (`bool`) – Передайте True, если администратор может закреплять сообщения, только для супергрупп

- `can_promote_members` (`bool`) – Передайте True, если администратор может добавлять новых администраторов с подмножеством его собственных прав администратора или понижать администраторов, которых он повысил, напрямую или косвенно (администраторами, которых он назначил)
- `is_anonymous` (`bool`) – Передайте True, если присутствие администратора в чате скрыто
- `can_manage_chat` (`bool`) – Передайте True, если администратор имеет доступ к логу событий чата, статистике чата, статистике сообщений в каналах, видеть участников канала, видеть анонимных администраторов в супергруппах и игнорировать медленный режим. Подразумевается любым другим правом администратора
- `can_manage_video_chats` (`bool`) – Передайте True, если администратор может управлять голосовыми чатами. На текущий момент, боты могут использовать это право администратора только для передачи другим администраторам.
- `can_manage_voice_chats` (`bool`) – Устарело, используйте can_manage_video_chats.
- `can_manage_topics` (`bool`) – Передайте True, если пользователю разрешено создавать, переименовывать, закрывать, и возобновлять топики, только для супергрупп
- `can_post_stories` (`bool`) – Pass True if the administrator can create the channel's stories
- `can_edit_stories` (`bool`) – Pass True if the administrator can edit the channel's stories
- `can_delete_stories` (`bool`) – Pass True if the administrator can delete the channel's stories

Результат
True в случае успеха.

Тип результата
`bool`

`register_business_connection_handler(`*callback: Callable*, *func: Callable | None = None*, *pass_bot: bool | None = False*, ***kwargs*`)`

Registers business connection handler.

Параметры

- `callback` (`function`) – функция-хендлер
- `func` (`function`) – Функция, используемая в качестве фильтра
- `pass_bot` (`bool`) – True, если вам нужно передать экземпляр класса TeleBot в хендлер(удобно для разбиения кода на файлы)
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`register_business_message_handler(`*callback: Callable*, *commands: List[str] | None = None*, *regexp: str | None = None*, *func: Callable | None = None*, *content_types: List[str] | None = None*, ***kwargs*`)`

Registers business connection handler.

Параметры

- `callback` (`function`) – функция-хендлер
- `commands` (`list` of `str`) – список команд
- `regexp` (`str`) – Регулярное выражение
- `func` (`function`) – Функция, используемая в качестве фильтра
- `content_types` (`list` of `str`) – Обрабатываемые виды контента. Обязан быть списком. По умолчанию [„text“]
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`register_callback_query_handler(`*callback: Callable*, *func: Callable*, *pass_bot: bool | None = False*, ***kwargs*`)`

Регистрирует хендлер callback query.

Параметры

- `callback` (`function`) – функция-хендлер
- `func` (`function`) – Функция, используемая в качестве фильтра
- `pass_bot` (`bool`) – True, если вам нужно передать экземпляр класса TeleBot в хендлер(удобно для разбиения кода на файлы)
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`register_channel_post_handler(`*callback: Callable*, *content_types: List[str] | None = None*, *commands: List[str] | None = None*, *regexp: str | None = None*, *func: Callable | None = None*, *pass_bot: bool | None = False*, ***kwargs*`)`

Регистрирует хендлер постов в каналах.

Параметры

- `callback` (`function`) – функция-хендлер
- `content_types` (`list` of `str`) – Обрабатываемые виды контента. Обязан быть списком. По умолчанию [„text“]
- `commands` (`list` of `str`) – список команд
- `regexp` (`str`) – Регулярное выражение
- `func` (`function`) – Функция, используемая в качестве фильтра
- `pass_bot` (`bool`) – True, если вам нужно передать экземпляр класса TeleBot в хендлер(удобно для разбиения кода на файлы)
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`register_chat_boost_handler(`*callback: Callable*, *func: Callable | None = None*, *pass_bot: bool | None = False*, ***kwargs*`)`

Registers chat boost handler.

Параметры

- `callback` (`function`) – функция-хендлер
- `func` (`function`) – Функция, используемая в качестве фильтра
- `pass_bot` – True, если вам нужно передать экземпляр класса TeleBot в хендлер(удобно для разбиения кода на файлы)
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`register_chat_join_request_handler(`*callback: Callable*, *func: Callable | None = None*, *pass_bot: bool | None = False*, ***kwargs*`)`

Регистрирует хендлер запросов на вступление в чат.

Параметры

- `callback` (`function`) – функция-хендлер
- `func` (`function`) – Функция, используемая в качестве фильтра
- `pass_bot` (`bool`) – True, если вам нужно передать экземпляр класса TeleBot в хендлер(удобно для разбиения кода на файлы)
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`register_chat_member_handler(`*callback: Callable*, *func: Callable | None = None*, *pass_bot: bool | None = False*, ***kwargs*`)`

Регистрирует хендлер смены состояний участников чата.

Параметры

- `callback` (`function`) – функция-хендлер
- `func` (`function`) – Функция, используемая в качестве фильтра
- `pass_bot` (`bool`) – True, если вам нужно передать экземпляр класса TeleBot в хендлер(удобно для разбиения кода на файлы)
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

:return:None

`register_chosen_inline_handler(`*callback: Callable*, *func: Callable*, *pass_bot: bool | None = False*, ***kwargs*`)`

Регистрирует хендлер выбора результата inline query.

Параметры

- `callback` (`function`) – функция-хендлер
- `func` (`function`) – Функция, используемая в качестве фильтра
- `pass_bot` (`bool`) – True, если вам нужно передать экземпляр класса TeleBot в хендлер(удобно для разбиения кода на файлы)

- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`register_deleted_business_messages_handler(`*callback: Callable, func: Callable | None = None, pass_bot: bool | None = False, **kwargs*`)`

Registers deleted business messages handler.

Параметры

- `callback` (`function`) – функция-хендлер
- `func` (`function`) – Функция, используемая в качестве фильтра
- `pass_bot` (`bool`) – True, если вам нужно передать экземпляр класса TeleBot в хендлер(удобно для разбиения кода на файлы)
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`register_edited_business_message_handler(`*callback: Callable, content_types: List[str] | None = None, commands: List[str] | None = None, regexp: str | None = None, func: Callable | None = None, pass_bot: bool | None = False, **kwargs*`)`

Registers edited message handler for business accounts.

Параметры

- `callback` (`function`) – функция-хендлер
- `content_types` (`list` of `str`) – Обрабатываемые виды контента. Обязан быть списком. По умолчанию [„text“]
- `commands` (`list` of `str`) – список команд
- `regexp` (`str`) – Регулярное выражение
- `func` (`function`) – Функция, используемая в качестве фильтра
- `pass_bot` (`bool`) – True, если вам нужно передать экземпляр класса TeleBot в хендлер(удобно для разбиения кода на файлы)
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`register_edited_channel_post_handler(`*callback: Callable, content_types: List[str] | None = None, commands: List[str] | None = None, regexp: str | None = None, func: Callable | None = None, pass_bot: bool | None = False, **kwargs*`)`

Регистрирует хендлер изменения постов в каналах.

Параметры

- `callback` (`function`) – функция-хендлер
- `content_types` (`list` of `str`) – Обрабатываемые виды контента. Обязан быть списком. По умолчанию [„text“]
- `commands` (`list` of `str`) – список команд

- `regexp` (`str`) – Регулярное выражение
- `func` (`function`) – Функция, используемая в качестве фильтра
- `pass_bot` (`bool`) – True, если вам нужно передать экземпляр класса TeleBot в хендлер(удобно для разбиения кода на файлы)
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
декорируемая функция

`register_edited_message_handler(`*callback: Callable, content_types: List[str] | None = None, commands: List[str] | None = None, regexp: str | None = None, func: Callable | None = None, chat_types: List[str] | None = None, pass_bot: bool | None = False, **kwargs*`)`

Регистрирует хендлер изменения сообщений.

Параметры

- `callback` (`function`) – функция-хендлер
- `content_types` (`list` of `str`) – Обрабатываемые виды контента. Обязан быть списком. По умолчанию [„text“]
- `commands` (`list` of `str`) – список команд
- `regexp` (`str`) – Регулярное выражение
- `func` (`function`) – Функция, используемая в качестве фильтра
- `chat_types` (`bool`) – True для приватных чатов
- `pass_bot` (`bool`) – True, если вам нужно передать экземпляр класса TeleBot в хендлер(удобно для разбиения кода на файлы)
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`register_for_reply(`*message:* Message, *callback: Callable, **args, **kwargs*`)` → None

Регистрирует функцию для вызова при получении ответа на выбранное сообщение.

Предупреждение: При использовании lambda функции в качестве *callback*, сохранение reply хендлеров не будет работать.

Параметры

- `message` (*`telebot.types.Message`*) – Сообщение, ответ на которое нужно ждать.
- `callback` (`Callable[[telebot.types.Message], None]`) – Функция, которую нужно вызвать при получении ответа на сообщение. Должна принимать параметр *message*, который будет содержать ответ на сообщение.
- `args` – Необязательные аргументы для вызываемой функции.
- `kwargs` – Необязательные именованные аргументы для вызываемой функции.

Результат
None

`register_for_reply_by_message_id(message_id: int, callback: Callable, *args, **kwargs)` → None

Регистрирует функцию для вызова при получении ответа на выбранное сообщение.

Предупреждение: При использовании lambda функции в качестве *callback*, сохранение reply хендлеров не будет работать.

Параметры

- `message_id` (`int`) – id сообщения, ответ на которое нужно ждать.
- `callback` (`Callable[[telebot.types.Message], None]`) – Функция, которую нужно вызвать при получении ответа на сообщение. Должна принимать параметр *message*, который будет содержать ответ на сообщение.
- `args` – Необязательные аргументы для вызываемой функции.
- `kwargs` – Необязательные именованные аргументы для вызываемой функции.

Результат

None

`register_inline_handler(callback: Callable, func: Callable, pass_bot: bool | None = False, **kwargs)`

Регистрирует хендлер inline query.

Параметры

- `callback` (`function`) – функция-хендлер
- `func` (`function`) – Функция, используемая в качестве фильтра
- `pass_bot` (`bool`) – True, если вам нужно передать экземпляр класса TeleBot в хендлер(удобно для разбиения кода на файлы)
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат

декорируемая функция

`register_message_handler(callback: Callable, content_types: List[str] | None = None, commands: List[str] | None = None, regexp: str | None = None, func: Callable | None = None, chat_types: List[str] | None = None, pass_bot: bool | None = False, **kwargs)`

Регистрирует хендлер сообщений.

Параметры

- `callback` (`function`) – функция-хендлер
- `content_types` (`list` of `str`) – Обрабатываемые виды контента. Обязан быть списком. По умолчанию [„text“]
- `commands` (`list` of `str`) – список команд
- `regexp` (`str`) – Регулярное выражение
- `func` (`function`) – Функция, используемая в качестве фильтра
- `chat_types` (`list` of `str`) – Список видов чатов
- `pass_bot` (`bool`) – True, если вам нужно передать экземпляр класса TeleBot в хендлер(удобно для разбиения кода на файлы)
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`register_message_reaction_count_handler(`*callback: Callable*, *func: Callable = None*, *pass_bot: bool | None = False*, ***kwargs*`)`

Registers message reaction count handler.

Параметры

- `callback` (`function`) – функция-хендлер
- `func` (`function`) – Функция, используемая в качестве фильтра
- `pass_bot` (`bool`) – True, если вам нужно передать экземпляр класса TeleBot в хендлер(удобно для разбиения кода на файлы)
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`register_message_reaction_handler(`*callback: Callable*, *func: Callable = None*, *pass_bot: bool | None = False*, ***kwargs*`)`

Registers message reaction handler.

Параметры

- `callback` (`function`) – функция-хендлер
- `func` (`function`) – Функция, используемая в качестве фильтра
- `pass_bot` (`bool`) – True, если вам нужно передать экземпляр класса TeleBot в хендлер(удобно для разбиения кода на файлы)
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`register_middleware_handler(`*callback*, *update_types=None*`)`

Добавляет функцию-middleware.

Эта функция зарегистрирует вашу функцию-middleware. Middleware функции исполняются до хендлеров. Будьте осторожны и проверяйте вид апдейта внутри функции, если указано более одного update_type

Пример:

bot = TeleBot(„TOKEN“)

bot.register_middleware_handler(print_channel_post_text, update_types=[„channel_post“, „edited_channel_post“])

Параметры

- `callback` (`function`) – Функция, которая будет использована в качестве middleware.
- `update_types` (`list` of `str`) – Необязательный список видов апдейтов, которые будут обработаны этим middleware хендлером.

Результат
None

`register_my_chat_member_handler(`*callback: Callable*, *func: Callable | None = None*, *pass_bot: bool | None = False*, ***kwargs*`)`

Регистрирует хендлер изменений статуса бота.

Параметры

- `callback` (`function`) – функция-хендлер
- `func` (`function`) – Функция, используемая в качестве фильтра
- `pass_bot` (`bool`) – True, если вам нужно передать экземпляр класса TeleBot в хендлер(удобно для разбиения кода на файлы)
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`register_next_step_handler(`*message:* Message, *callback: Callable*, **args*, ***kwargs*`)` → None

Регистрирует функцию для вызова при получении нового сообщения после указанного.

Предупреждение: При использовании lambda функции в качестве *callback*, сохранение next step хендлеров не будет работать.

Параметры

- `message` (*`telebot.types.Message`*) – Сообщение, после которого нужно обработать следующее в том же чате.
- `callback` (`Callable[[telebot.types.Message], None]`) – Функция для вызова при получении нового сообщения.
- `args` – Аргументы для передачи в функцию
- `kwargs` – Аргументы для передачи в функцию

Результат
None

`register_next_step_handler_by_chat_id(`*chat_id: int*, *callback: Callable*, **args*, ***kwargs*`)` → None

Регистрирует функцию для вызова при получении нового сообщения в заданном чате.

Предупреждение: При использовании lambda функции в качестве *callback*, сохранение next step хендлеров не будет работать.

Параметры

- `chat_id` (`int`) – Чат (id чата), в котором нужно обработать новое сообщение.
- `callback` (`Callable[[telebot.types.Message], None]`) – Функция для вызова при получении нового сообщения.
- `args` – Аргументы для передачи в функцию
- `kwargs` – Аргументы для передачи в функцию

Результат
None

`register_poll_answer_handler(`*callback: Callable*, *func: Callable*, *pass_bot: bool | None = False*, ***kwargs*`)`

Регистрирует хендлер ответов в опросах.

Параметры

- `callback` (`function`) – функция-хендлер
- `func` (`function`) – Функция, используемая в качестве фильтра
- `pass_bot` (`bool`) – True, если вам нужно передать экземпляр класса TeleBot в хендлер(удобно для разбиения кода на файлы)
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`register_poll_handler(`*callback: Callable*, *func: Callable*, *pass_bot: bool | None = False*, ***kwargs*`)`

Регистрирует хендлер изменений состояния опросов.

Параметры

- `callback` (`function`) – функция-хендлер
- `func` (`function`) – Функция, используемая в качестве фильтра
- `pass_bot` (`bool`) – True, если вам нужно передать экземпляр класса TeleBot в хендлер(удобно для разбиения кода на файлы)
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`register_pre_checkout_query_handler(`*callback: Callable*, *func: Callable*, *pass_bot: bool | None = False*, ***kwargs*`)`

Регистрирует хендлер pre-checkout query.

Параметры

- `callback` (`function`) – функция-хендлер
- `func` – Функция, используемая в качестве фильтра
- `pass_bot` (`bool`) – True, если вам нужно передать экземпляр класса TeleBot в хендлер(удобно для разбиения кода на файлы)
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
декорируемая функция

`register_removed_chat_boost_handler(`*callback: Callable*, *func: Callable | None = None*, *pass_bot: bool | None = False*, ***kwargs*`)`

Registers removed chat boost handler.

Параметры

- `callback` (`function`) – функция-хендлер
- `func` (`function`) – Функция, используемая в качестве фильтра
- `pass_bot` – True, если вам нужно передать экземпляр класса TeleBot в хендлер(удобно для разбиения кода на файлы)
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`register_shipping_query_handler(callback: Callable, func: Callable, pass_bot: bool | None = False, **kwargs)`

Регистрирует хендлер shipping query.

Параметры

- `callback` (`function`) – функция-хендлер
- `func` (`function`) – Функция, используемая в качестве фильтра
- `pass_bot` (`bool`) – True, если вам нужно передать экземпляр класса TeleBot в хендлер(удобно для разбиения кода на файлы)
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`remove_webhook() → bool`

Удаляет вебхук, используя set_webhook().

Результат
True в случае успеха.

Тип результата
`bool`

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

Handles new incoming chat boost state. it passes `telebot.types.ChatBoostRemoved` object.

Параметры

- `func` (`function`) – Функция, используемая в качестве фильтра
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`reopen_forum_topic(chat_id: str | int, message_thread_id: int) → bool`

Используйте этот метод, чтобы возобновить закрытый топик в супергруппе с топиками. Бот должен быть администратором чата и иметь права администратора can_manage_topics, кроме случаев, когда бот является создателем топика. Возвращает True в случае успеха.

Документация Telegram: https://core.telegram.org/bots/api#reopenforumtopic

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `message_thread_id` (`int`) – id топика для возобновления

Результат
В случае успеха возвращается True.

Тип результата
`bool`

`reopen_general_forum_topic(chat_id: int | str) → bool`

Используйте этот метод, чтобы возобновить топик „General“ в супергруппе с топиками. Бот должен быть администратором чата и иметь права администратора can_manage_topics, кроме случаев, когда бот является создателем топика. Возвращает True в случае успеха.

Документация Telegram: https://core.telegram.org/bots/api#reopengeneralforumtopic

Параметры
chat_id (int or str) – Уникальный id чата или username канала (в формате @channelusername)

replace_sticker_in_set(*user_id: int*, *name: str*, *old_sticker: str*, *sticker:* InputSticker) → bool

Use this method to replace an existing sticker in a sticker set with a new one. The method is equivalent to calling deleteStickerFromSet, then addStickerToSet,
then setStickerPositionInSet. Returns True on success.

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

Параметры

- user_id (int) – User identifier of the sticker set owner
- name (str) – Имя стикерпака
- old_sticker (str) – File identifier of the replaced sticker
- sticker (*telebot.types.InputSticker*) – A JSON-serialized object with information about the added sticker. If exactly the same sticker had already been added to the set, then the set remains unchanged.

Результат
Возвращает True в случае успеха.

Тип результата
bool

reply_to(*message:* Message, *text: str*, ***kwargs*) → *Message*

Convenience function for *send_message(message.chat.id, text, reply_parameters=(message.message_id...), **kwargs)*

Параметры

- message (*types.Message*) – Экземпляр класса *telebot.types.Message*
- text (str) – Текст сообщения.
- kwargs – Дополнительные именованные аргументы, передаваемые в *telebot.TeleBot.send_message()*

Результат
В случае успеха возвращает отправленное сообщение (Message).

Тип результата
telebot.types.Message

reset_data(*user_id: int*, *chat_id: int | None = None*)

Сбросить данные о пользователе в чате.

Параметры

- user_id (int) – id пользователя
- chat_id (int) – id чата

Результат
None

`restrict_chat_member(`*chat_id: int | str*, *user_id: int*, *until_date: int | datetime | None = None*, *can_send_messages: bool | None = None*, *can_send_media_messages: bool | None = None*, *can_send_polls: bool | None = None*, *can_send_other_messages: bool | None = None*, *can_add_web_page_previews: bool | None = None*, *can_change_info: bool | None = None*, *can_invite_users: bool | None = None*, *can_pin_messages: bool | None = None*, *permissions:* ChatPermissions *| None = None*, *use_independent_chat_permissions: bool | None = None*`)` → bool

Используйте этот метод, чтобы ограничить пользователя в супергруппе. Бот должен быть администратором супергруппы и иметь соответствующие права администратора. Передайте True во все boolean параметры, чтобы снять с пользователя ограничения.

Документация Telegram: https://core.telegram.org/bots/api#restrictchatmember

> **Предупреждение:** Individual parameters are deprecated and will be removed, use „permissions“ instead.

Параметры

- `chat_id` (`int` or `str`) – Уникальный id группы или username супергруппы или канала (в формате @channelusername)
- `user_id` (`int`) – Уникальный id сделавшего запрос пользователя
- `until_date` (`int` or `datetime`, optional) – Дата, когда ограничения будут сняты с пользователя, UNIX timestamp. Если пользователь ограничен более чем на 366 дней или менее чем на 30 секунд с текущего момента, он будет ограничен навсегда (пока ограничения не будут сняты вручную)
- `can_send_messages` (`bool`) – deprecated
- `can_send_media_messages` (`bool`) – deprecated
- `can_send_polls` (`bool`) – deprecated
- `can_send_other_messages` (`bool`) – deprecated
- `can_add_web_page_previews` (`bool`) – deprecated
- `can_change_info` (`bool`) – deprecated
- `can_invite_users` (`bool`) – deprecated
- `can_pin_messages` (`bool`) – deprecated
- `use_independent_chat_permissions` (`bool`, optional) – Pass True if chat permissions are set independently. Otherwise, the can_send_other_messages and can_add_web_page_previews permissions will imply the can_send_messages, can_send_audios, can_send_documents, can_send_photos, can_send_videos, can_send_video_notes, and can_send_voice_notes permissions; the can_send_polls permission will imply the can_send_messages permission.
- `permissions` (*`telebot.types.ChatPermissions`*) – ChatPermissions object defining permissions.

Результат

True в случае успеха

Тип результата

`bool`

`retrieve_data(user_id: int, chat_id: int | None = None)` → Any | None

Возвращает контекстный менеджер с данными о пользователе в чате.

Параметры

- `user_id` (*int*) – id пользователя
- `chat_id` (*int, optional*) – Уникальный id чата, по умолчанию user_id

Результат
Контекстный менеджер с данными о пользователе в чате.

Тип результата
Optional[Any]

`revoke_chat_invite_link(chat_id: int | str, invite_link: str)` → *ChatInviteLink*

Используйте этот метод, чтобы аннулировать ссылку-приглашение, созданную ботом. Примечание: Если аннулируется главная ссылка-приглашение, автоматически генерируется новая. Бот должен быть администратором чата и иметь соответствующие права администратора.

Документация Telegram: https://core.telegram.org/bots/api#revokechatinvitelink

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `invite_link` (`str`) – Ссылка-приглашение, которую нужно аннулировать

Результат
Возвращает новую ссылку-приглашение (ChatInviteLink).

Тип результата
telebot.types.ChatInviteLink

`run_webhooks(listen: str | None = '127.0.0.1', port: int | None = 443, url_path: str | None = None, certificate: str | None = None, certificate_key: str | None = None, webhook_url: str | None = None, max_connections: int | None = None, allowed_updates: List | None = None, ip_address: str | None = None, drop_pending_updates: bool | None = None, timeout: int | None = None, secret_token: str | None = None, secret_token_length: int | None = 20)`

Этот класс устанавливает вебхуки и мониторит указанный URL и порт.

Требует fastapi, uvicorn и последнюю версию starlette.

Параметры

- `listen` (`str`, optional) – IP адрес для мониторинга, по умолчанию «127.0.0.1»
- `port` (`int`, optional) – Порт, который будет использован для мониторинга вебхуков. По умолчанию 443
- `url_path` (`str`, optional) – Путь к вебхуку(по умолчанию /token). По умолчанию None
- `certificate` (`str`, optional) – Путь к файлу с SSL сертификатом, по умолчанию None
- `certificate_key` (`str`, optional) – Путь к файлу с приватным ключом SSL сертификата, по умолчанию None
- `webhook_url` (`str`, optional) – URL вебхука, по умолчанию None

- `max_connections` (`int`, optional) – Максимально-допустимое количество одновременных HTTPS подключений к вебхуку для доставки апдейтов, 1-100. По умолчанию 40. Используйте меньшие значения, чтобы уменьшить нагрузку на ваш сервер и большие значения для увеличения пропускной способности вашего бота, по умолчанию None.
- `allowed_updates` (`list`, optional) – Список видов апдейтов, которые вы хотите получать, в формате JSON. Например, укажите ["message", "edited_channel_post", "callback_query"], чтобы получать апдейты только этих видов. Полный список доступных видов апдейтов - util.update_types. Укажите пустой список, чтобы получать все апдейты, кроме chat_member (по умолчанию). Если не задан, будет использована последняя настройка. По умолчанию None
- `ip_address` (`str`, optional) – Фиксированный IP адрес, который будет использоваться для отправки запросов к вебхуку вместо IP адреса, полученного через DNS, по умолчанию None
- `drop_pending_updates` (`bool`, optional) – Передайте True, чтобы удалить все предшествующие запуску бота апдейты, по умолчанию None
- `timeout` (`int`, optional) – Тайм-аут запроса, по умолчанию None
- `secret_token` (`str`, optional) – Секретный токен для верификации запроса к вебхуку, по умолчанию None
- `secret_token_length` (`int`, optional) – Длина секретного токена, по умолчанию 20

Исключение

`ImportError` – Если необходимые библиотеки не были установлены.

`send_animation`(*chat_id: int | str*, *animation: Any | str*, *duration: int | None = None*, *width: int | None = None*, *height: int | None = None*, *thumbnail: str | Any | None = None*, *caption: str | None = None*, *parse_mode: str | None = None*, *caption_entities: List[MessageEntity] | None = None*, *disable_notification: bool | None = None*, *protect_content: bool | None = None*, *reply_to_message_id: int | None = None*, *allow_sending_without_reply: bool | None = None*, *reply_markup: InlineKeyboardMarkup | ReplyKeyboardMarkup | ReplyKeyboardRemove | ForceReply | None = None*, *timeout: int | None = None*, *message_thread_id: int | None = None*, *has_spoiler: bool | None = None*, *thumb: str | Any | None = None*, *reply_parameters: ReplyParameters | None = None*, *business_connection_id: str | None = None*) → *Message*

Используйте этот метод, чтобы отправить гифку (GIF или H.264/MPEG-4 AVC видео без звука). В случае успеха возвращается отправленное сообщение (Message). На текущий момент, боты могут отправлять гифки весом до 50 MB, это ограничение может измениться в будущем.

Документация Telegram: https://core.telegram.org/bots/api#sendanimation

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `animation` (`str` or *`telebot.types.InputFile`*) – Гиф-ка для отправки. Передайте file_id (String), чтобы отправить гифку, которая уже загружена на сервера Telegram (рекомендуется), передайте HTTP URL (String), чтобы отправить гифку из интернета или загрузите новую гифку с помощью multipart/form-data.

- `duration` (`int`) – Длительность отправленной гифки в секундах
- `width` (`int`) – Ширина гифки
- `height` (`int`) – Высота гифки
- `thumbnail` (`str` or *telebot.types.InputFile*) – Обложка отправленного файла; может быть проигнорирована, если генерация обложки поддерживается на стороне сервера. Обложка должна быть картинкой в формате JPEG и весить менее 200 kB. Ширина и высота обложки не должны превышать 320. Игнорируется, если файл не загружен с помощью multipart/form-data. Обложки не могут быть использованы повторно и могут быть загружены только как новый файл, так что вы можете передать "attach://<file_attach_name>" если обложка была загружена с помощью multipart/form-data под именем <file_attach_name>.
- `caption` (`str`) – Подпись к гифке (может быть использована при повторной отправке гифки по file_id), 0-1024 символа после форматирования
- `parse_mode` (`str`) – Режим форматирования подписи к гифке
- `protect_content` (`bool`) – Запретить пересылку и сохранение содержимого сообщения
- `reply_to_message_id` (`int`) – deprecated.
- `allow_sending_without_reply` (`bool`) – deprecated.
- `reply_markup` (*telebot.types.InlineKeyboardMarkup* or *telebot.types.ReplyKeyboardMarkup* or *telebot.types.ReplyKeyboardRemove* or *telebot.types.ForceReply*) – Дополнительные элементы интерфейса. Inline клавиатура, текстовая клавиатура, запрос на удаление текстовой клавиатуры или запрос на ответ от пользователя.
- `disable_notification` (`bool`) – Отправить сообщение, при получении которого пользователи получат уведомление без звука.
- `timeout` (`int`) – Таймаут запроса в секундах.
- `caption_entities` (`list` of *telebot.types.MessageEntity*) – Список отформатированных частей подписи, можно использовать вместо parse_mode
- `message_thread_id` (`int`) – id топика, в который будет отправлено видео
- `has_spoiler` (`bool`) – Передайте True, если гифку нужно отправить как спойлер
- `thumb` (`str` or *telebot.types.InputFile*) – deprecated.
- `reply_parameters` (*telebot.types.ReplyParameters*) – Reply parameters.
- `business_connection_id` (`str`) – Identifier of a business connection

Результат

В случае успеха возвращает отправленное сообщение (Message).

Тип результата

telebot.types.Message

send_audio(*chat_id: int | str*, *audio: Any | str*, *caption: str | None = None*, *duration: int | None = None*, *performer: str | None = None*, *title: str | None = None*, *reply_to_message_id: int | None = None*, *reply_markup:* InlineKeyboardMarkup | ReplyKeyboardMarkup | ReplyKeyboardRemove | ForceReply | *None = None*, *parse_mode: str | None = None*, *disable_notification: bool | None = None*, *timeout: int | None = None*, *thumbnail: str | Any | None = None*, *caption_entities: List[*MessageEntity*] | None = None*, *allow_sending_without_reply: bool | None = None*, *protect_content: bool | None = None*, *message_thread_id: int | None = None*, *thumb: str | Any | None = None*, *reply_parameters:* ReplyParameters | *None = None*, *business_connection_id: str | None = None*) → *Message*

Используйте этот метод, чтобы отправить аудио, если вы хотите, чтобы клиенты (приложения) Telegram проигрывали их в музыкальном проигрывателе. Ваше аудио должно быть в формате .MP3 или .M4A. В случае успеха возвращается отправленное сообщение (Message). На текущий момент, боты могут отправлять аудио весом до 50 MB, это ограничение может измениться в будущем.

Для отправки голосовых сообщений, используйте метод send_voice

Документация Telegram: https://core.telegram.org/bots/api#sendaudio

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `audio` (`str` or *`telebot.types.InputFile`*) – Аудио для отправки. Передайте file_id (String), чтобы отправить аудио, которое уже загружено на сервера Telegram (рекомендуется), передайте HTTP URL (String), чтобы отправить аудио из интернета или загрузите новое с помощью multipart/form-data. Аудио должно быть в формате .MP3 или .M4A.
- `caption` (`str`) – Подпись к аудио, 0-1024 символа после форматирования
- `duration` (`int`) – Длительность аудио в секундах
- `performer` (`str`) – Исполнитель
- `title` (`str`) – Название трека
- `reply_markup` (*`telebot.types.InlineKeyboardMarkup`* or *`telebot.types.ReplyKeyboardMarkup`* or *`telebot.types.ReplyKeyboardRemove`* or *`telebot.types.ForceReply`*)
- `parse_mode` (`str`) – Режим форматирования подписи к аудио. См. formatting options для получения подробностей.
- `disable_notification` (`bool`) – Отправить сообщение, при получении которого пользователи получат уведомление без звука.
- `reply_to_message_id` (`int`) – deprecated.
- `allow_sending_without_reply` (`bool`) – deprecated.
- `timeout` (`int`) – Таймаут запроса в секундах.
- `thumbnail` (`str` or *`telebot.types.InputFile`*) – Обложка отправленного файла; может быть проигнорирована, если генерация обложки поддерживается на стороне сервера. Обложка должна быть картинкой в формате JPEG и весить менее 200 kB. Ширина и высота обложки не должны превышать 320. Игнорируется, если файл не загружен с помощью multipart/form-data. Обложки не могут быть использованы повторно и могут быть загружены только как новый файл, так что вы можете передать "attach://<file_attach_name>"

если обложка была загружена с помощью multipart/form-data под именем <file_attach_name>.

- `caption_entities` (`list` of *telebot.types.MessageEntity*) – Список отформатированных частей подписи в формате JSON, можно использовать вместо parse_mode
- `protect_content` (`bool`) – Запретить пересылку и сохранение содержимого сообщения
- `message_thread_id` (`int`) – id топика, в который нужно отправить сообщение
- `thumb` (`str` or *telebot.types.InputFile*) – deprecated.
- `reply_parameters` (*telebot.types.ReplyParameters*) – Reply parameters.
- `business_connection_id` (`str`) – Identifier of a business connection, in which the message will be sent

Результат
В случае успеха возвращает отправленное сообщение (Message).

Тип результата
telebot.types.Message

`send_chat_action`(*chat_id: int | str*, *action: str*, *timeout: int | None = None*, *message_thread_id: int | None = None*, *business_connection_id: str | None = None*) → bool

Используйте этот метод, когда вам нужно показать пользователю, что бот что-то делает. Статус устанавливается на 5 секунд или менее (когда от бота приходит сообщение, клиенты (приложения) Telegram убирают статус typing). Возвращает True в случае успеха.

Пример: ImageBot-у требуется время, чтобы обработать запрос и загрузить изображение. Вместо отправки текстового сообщения "Отправка изображения, пожалуйста подождите...", бот может использовать sendChatAction с параметром action = upload_photo. Пользователь увидит статус бота "sending photo".

Документация Telegram: https://core.telegram.org/bots/api#sendchataction

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала
- `action` (`str`) – Тип действия. Выберите один, в зависимости от того, что получит пользователь: typing для текстовых сообщений, upload_photo для фото, record_video или upload_video для видео, record_voice или upload_voice для голосовых сообщений, upload_document для файлов, choose_sticker для стикеров, find_location для данных о местоположении, record_video_note или upload_video_note для видео заметок (кружочков).
- `timeout` (`int`) – Таймаут запроса в секундах.
- `message_thread_id` (`int`) – id топика, на сообщение из которого нужно ответить(только для супергрупп)
- `business_connection_id` (`str`) – Identifier of a business connection

Результат
Возвращает True в случае успеха.

Тип результата
`bool`

`send_contact(`*chat_id: int | str*, *phone_number: str*, *first_name: str*, *last_name: str | None = None*, *vcard: str | None = None*, *disable_notification: bool | None = None*, *reply_to_message_id: int | None = None*, *reply_markup:* InlineKeyboardMarkup | ReplyKeyboardMarkup | ReplyKeyboardRemove | ForceReply | *None = None*, *timeout: int | None = None*, *allow_sending_without_reply: bool | None = None*, *protect_content: bool | None = None*, *message_thread_id: int | None = None*, *reply_parameters:* ReplyParameters | *None = None*, *business_connection_id: str | None = None*`)` → *Message*

Используйте этот метод, чтобы отправить контакт. В случае успеха возвращается отправленное сообщение (Message).

Документация Telegram: https://core.telegram.org/bots/api#sendcontact

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала
- `phone_number` (`str`) – Телефонный номер контакта
- `first_name` (`str`) – Имя контакта
- `last_name` (`str`) – Фамилия контакта
- `vcard` (`str`) – Дополнительные данные о контакте в формате vCard, 0-2048 байт
- `disable_notification` (`bool`) – Отправить сообщение, при получении которого пользователи получат уведомление без звука.
- `reply_to_message_id` (`int`) – deprecated.
- `allow_sending_without_reply` (`bool`) – deprecated.
- `reply_markup` (*`telebot.types.InlineKeyboardMarkup`* or *`telebot.types.ReplyKeyboardMarkup`* or *`telebot.types.ReplyKeyboardRemove`* or *`telebot.types.ForceReply`*) – Дополнительные элементы интерфейса. Inline клавиатура, текстовая клавиатура, запрос на удаление текстовой клавиатуры или запрос на ответ от пользователя.
- `timeout` (`int`) – Таймаут запроса в секундах.
- `protect_content` (`bool`) – Запретить пересылку и сохранение содержимого сообщения
- `message_thread_id` (`int`) – id топика, на сообщение из которого нужно ответить
- `reply_parameters` (*`telebot.types.ReplyParameters`*) – Reply parameters.
- `business_connection_id` (`str`) – Identifier of a business connection

Результат

В случае успеха возвращает отправленное сообщение (Message).

Тип результата

`telebot.types.Message`

`send_dice(`*chat_id: int | str*, *emoji: str | None = None*, *disable_notification: bool | None = None*, *reply_to_message_id: int | None = None*, *reply_markup:* InlineKeyboardMarkup | ReplyKeyboardMarkup | ReplyKeyboardRemove | ForceReply | *None = None*, *timeout: int | None = None*, *allow_sending_without_reply: bool | None = None*, *protect_content: bool | None = None*, *message_thread_id: int | None = None*, *reply_parameters:* ReplyParameters | *None = None*, *business_connection_id: str | None = None*`)` → *Message*

Используйте этот метод, чтобы отправить анимированный эмодзи, который покажет случайное значение. В случае успеха возвращается отправленное сообщение (Message).

Документация Telegram: https://core.telegram.org/bots/api#senddice

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `emoji` (`str`) – Эмодзи, на котором основана анимация. На текущий момент, должно быть одним из “”, “”, “”, “”, “”, или “”. Значение может быть 1-6 для “”, “” и “”, 1-5 для “” и “”, и 1-64 для “”. По умолчанию “”
- `disable_notification` (`bool`) – Отправить сообщение, при получении которого пользователи получат уведомление без звука.
- `reply_markup` (*`telebot.types.InlineKeyboardMarkup`* or *`telebot.types.ReplyKeyboardMarkup`* or *`telebot.types.ReplyKeyboardRemove`* or *`telebot.types.ForceReply`*) – Дополнительные элементы интерфейса. Inline клавиатура, текстовая клавиатура, запрос на удаление текстовой клавиатуры или запрос на ответ от пользователя.
- `reply_to_message_id` (`int`) – deprecated.
- `allow_sending_without_reply` (`bool`) – deprecated.
- `timeout` (`int`) – Таймаут запроса в секундах.
- `protect_content` (`bool`) – Запретить пересылку и сохранение содержимого сообщения
- `message_thread_id` (`int`) – id топика, в который нужно отправить сообщение
- `reply_parameters` (*`telebot.types.ReplyParameters`*) – Additional parameters for replies to messages
- `business_connection_id` (`str`) – Identifier of a business connection, in which the message will be sent

Результат

В случае успеха возвращает отправленное сообщение (Message).

Тип результата

`telebot.types.Message`

`send_document`(*chat_id: int | str*, *document: Any | str*, *reply_to_message_id: int | None = None*, *caption: str | None = None*, *reply_markup:* InlineKeyboardMarkup | ReplyKeyboardMarkup | ReplyKeyboardRemove | ForceReply | *None = None*, *parse_mode: str | None = None*, *disable_notification: bool | None = None*, *timeout: int | None = None*, *thumbnail: str | Any | None = None*, *caption_entities: List[*MessageEntity*] | None = None*, *allow_sending_without_reply: bool | None = None*, *visible_file_name: str | None = None*, *disable_content_type_detection: bool | None = None*, *data: str | Any | None = None*, *protect_content: bool | None = None*, *message_thread_id: int | None = None*, *thumb: str | Any | None = None*, *reply_parameters:* ReplyParameters | *None = None*, *business_connection_id: str | None = None*) → *Message*

Используйте этот метод, чтобы отправить файл.

Документация Telegram: https://core.telegram.org/bots/api#senddocument

Параметры

- chat_id (int or str) – Уникальный id чата или username канала (в формате @channelusername)
- document (str or *telebot.types.InputFile*) – (документ) Файл для отправки. Передайте file_id (String), чтобы отправить файл, который уже загружен на сервера Telegram (рекомендуется), передайте HTTP URL (String), чтобы отправить файл из интернета или загрузите новый с помощью multipart/form-data
- caption (str) – Подпись к файлу (может быть использована при повторной отправке файла по file_id), 0-1024 символа после форматирования
- reply_markup (*telebot.types.InlineKeyboardMarkup* or *telebot.types.ReplyKeyboardMarkup* or *telebot.types.ReplyKeyboardRemove* or *telebot.types.ForceReply*) – Дополнительные элементы интерфейса. Inline клавиатура, текстовая клавиатура, запрос на удаление текстовой клавиатуры или запрос на ответ от пользователя.
- parse_mode (str) – Режим форматирования частей подписи к файлу
- disable_notification (bool) – Отправить сообщение, при получении которого пользователи получат уведомление без звука.
- reply_to_message_id (int) – deprecated.
- allow_sending_without_reply (bool) – deprecated.
- timeout (int) – Таймаут запроса в секундах.
- thumbnail (str or *telebot.types.InputFile*) – InputFile или String : Обложка отправленного файла; может быть проигнорирована, если генерация обложки поддерживается на стороне сервера. Обложка должна быть картинкой в формате JPEG и весить менее 200 kB. Ширина и высота обложки не должны превышать 320. Игнорируется, если файл не загружен с помощью multipart/form-data. Обложки не могут быть использованы повторно и могут быть загружены только как новый файл, так что вы можете передать "attach://<file_attach_name>" если обложка была загружена с помощью multipart/form-data под именем <file_attach_name>.
- caption_entities (list of *telebot.types.MessageEntity*) – Список отформатированных частей подписи в формате JSON, можно использовать вместо parse_mode
- visible_file_name (str) – позволяет задать имя файла, которое будет показано в Telegram вместо настоящего
- disable_content_type_detection (bool) – Отключает автоматическое обнаружение типа файла на стороне сервера для файлов, загруженных с помощью multipart/form-data
- data (str) – опечатка: не используйте
- protect_content (bool) – Запретить пересылку и сохранение содержимого сообщения
- message_thread_id (int) – Топик, в которой сообщение будет отправлено
- thumb (str or *telebot.types.InputFile*) – deprecated.
- reply_parameters (*telebot.types.ReplyParameters*) – Reply parameters.
- business_connection_id (str) – Identifier of a business connection, in which the message will be sent

Результат
В случае успеха возвращает отправленное сообщение (Message).

Тип результата
telebot.types.Message

`send_game(chat_id: int | str, game_short_name: str, disable_notification: bool | None = None, reply_to_message_id: int | None = None, reply_markup: InlineKeyboardMarkup | ReplyKeyboardMarkup | ReplyKeyboardRemove | ForceReply | None = None, timeout: int | None = None, allow_sending_without_reply: bool | None = None, protect_content: bool | None = None, message_thread_id: int | None = None, reply_parameters: ReplyParameters | None = None, business_connection_id: str | None = None) → Message`

Используется для отправки игры.

Документация Telegram: https://core.telegram.org/bots/api#sendgame

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `game_short_name` (`str`) – Короткое имя игры, служит в качестве уникального id игры. Настройте свои игрычерез @BotFather.
- `disable_notification` (`bool`) – Отправить сообщение, при получении которого пользователи получат уведомление без звука.
- `reply_to_message_id` (`int`) – deprecated. If the message is a reply, ID of the original message
- `allow_sending_without_reply` (`bool`) – deprecated. Pass True, if the message should be sent even if the specified replied-to message is not found
- `reply_markup` (`InlineKeyboardMarkup` or `ReplyKeyboardMarkup` or `ReplyKeyboardRemove` or `ForceReply`) – Дополнительные элементы интерфейса. Inline клавиатура, текстовая клавиатура, запрос на удаление текстовой клавиатуры или запрос на ответ от пользователя.
- `timeout` (`int`) – Тайм-аут в секундах, ожидание ответа от бота.
- `protect_content` (`bool`) – Запретить пересылку и сохранение содержимого сообщения
- `message_thread_id` (`int`) – id топика, в который будет отправлено сообщение с игрой.
- `reply_parameters` (`ReplyParameters`) – Reply parameters
- `business_connection_id` (`str`) – Unique identifier of the business connection

Результат
В случае успеха возвращает отправленное сообщение (Message).

Тип результата
types.Message

`send_invoice`(*chat_id: int | str*, *title: str*, *description: str*, *invoice_payload: str*, *provider_token: str*, *currency: str*, *prices: List[LabeledPrice]*, *start_parameter: str | None = None*, *photo_url: str | None = None*, *photo_size: int | None = None*, *photo_width: int | None = None*, *photo_height: int | None = None*, *need_name: bool | None = None*, *need_phone_number: bool | None = None*, *need_email: bool | None = None*, *need_shipping_address: bool | None = None*, *send_phone_number_to_provider: bool | None = None*, *send_email_to_provider: bool | None = None*, *is_flexible: bool | None = None*, *disable_notification: bool | None = None*, *reply_to_message_id: int | None = None*, *reply_markup: InlineKeyboardMarkup | ReplyKeyboardMarkup | ReplyKeyboardRemove | ForceReply | None = None*, *provider_data: str | None = None*, *timeout: int | None = None*, *allow_sending_without_reply: bool | None = None*, *max_tip_amount: int | None = None*, *suggested_tip_amounts: List[int] | None = None*, *protect_content: bool | None = None*, *message_thread_id: int | None = None*, *reply_parameters: ReplyParameters | None = None*) → *Message*

Отправляет инвойс.

Документация Telegram: https://core.telegram.org/bots/api#sendinvoice

Параметры

- `chat_id` (`int` or `str`) – Уникальный id приватного чата
- `title` (`str`) – Название товара, 1-32 символа
- `description` (`str`) – Описание товара, 1-255 символов
- `invoice_payload` (`str`) – Дополнительные данные, 1-128 байт. Не будет показано пользователю, используйте во внутренних процессах.
- `provider_token` (`str`) – Токен платежной системы, полученный через @BotFather
- `currency` (`str`) – Трехбуквенный код валюты в формате ISO 4217, см. https://core.telegram.org/bots/payments#supported-currencies
- `prices` (List[*`types.LabeledPrice`*]) – Детали цены, список компонент (например цена продукта, налог, скидка, стоимость доставки, налог на доставку, бонус и т.д.)
- `start_parameter` (`str`) – Уникальный deep-linking параметр, который может быть использован для генерации этого инвойса при использовании в качестве параметра /start
- `photo_url` (`str`) – URL фото продукта. Может быть фото товаров или рекламным изображением сервиса. Людям больше нравится, когда они видят, за что платят.
- `photo_size` (`int`) – Вес изображения в байтах
- `photo_width` (`int`) – Ширина изображения
- `photo_height` (`int`) – Высота изображения
- `need_name` (`bool`) – Передайте True, если для совершения заказа требуется полное имя пользователя
- `need_phone_number` (`bool`) – Передайте True, если для совершения заказа требуется номер телефона пользователя
- `need_email` (`bool`) – Передайте True, если для совершения заказа требуется email пользователя

- `need_shipping_address` (`bool`) – Передайте True, если для совершения заказа требуется адрес доставки
- `is_flexible` (`bool`) – Передайте True, если окончательная цена зависит от способа доставки
- `send_phone_number_to_provider` (`bool`) – Передайте True, если номер телефона пользователя нужно отправить платежной системе
- `send_email_to_provider` (`bool`) – Передайте True, если email пользователя нужно отправить платежной системе
- `disable_notification` (`bool`) – Отправить сообщение, при получении которого пользователи получат уведомление без звука.
- `reply_to_message_id` (`int`) – deprecated. If the message is a reply, ID of the original message
- `allow_sending_without_reply` (`bool`) – deprecated. Pass True, if the message should be sent even if the specified replied-to message is not found
- `reply_markup` (`str`) – JSON-сериализованный объект inline клавиатуры. Если пустой, будет показана одна кнопка „Pay total price“. Если не пустой, первая кнопка должна быть кнопкой для оплаты
- `provider_data` (`str`) – Данные о инвойсе в формате JSON, которые будут переданы платежной системе. Подробное описание обязательных полей должно быть предоставлено провайдером платежной системы.
- `timeout` (`int`) – Тайм-аут запроса, по умолчанию None
- `max_tip_amount` (`int`) – Максимальный размер чаевых в наименьших единицах выбранной валюты
- `suggested_tip_amounts` (`list` of `int`) – Массив предлагаемых вариантов чаевых в наименьших единицах выбранной валюты в формате JSON. Можно задать не более 4 вариантов. Варианты чаевых должны быть больше нуля, перечисленные в порядке строгого возрастания и не превышать max_tip_amount.
- `protect_content` (`bool`) – Запретить пересылку и сохранение содержимого сообщения
- `message_thread_id` (`int`) – id топика, в который будет отправлен инвойс
- `reply_parameters` (*`types.ReplyParameters`*) – Required if the message is a reply. Additional interface options.

Результат

В случае успеха возвращает отправленное сообщение (Message).

Тип результата

`types.Message`

`send_location`(*chat_id: int | str*, *latitude: float*, *longitude: float*, *live_period: int | None = None*, *reply_to_message_id: int | None = None*, *reply_markup:* InlineKeyboardMarkup | ReplyKeyboardMarkup | ReplyKeyboardRemove | ForceReply | *None = None*, *disable_notification: bool | None = None*, *timeout: int | None = None*, *horizontal_accuracy: float | None = None*, *heading: int | None = None*, *proximity_alert_radius: int | None = None*, *allow_sending_without_reply: bool | None = None*, *protect_content: bool | None = None*, *message_thread_id: int | None = None*, *reply_parameters:* ReplyParameters | *None = None*, *business_connection_id: str | None = None*) → *Message*

Используйте этот метод, чтобы отправить точку на карте. В случае успеха возвращается отправленное сообщение (Message).

Документация Telegram: https://core.telegram.org/bots/api#sendlocation

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `latitude` (`float`) – Широта
- `longitude` (`float`) – Долгота
- `live_period` (`int`) – Время в секундах, в течение которого местоположение будет обновляться (см. Live Locations), должно быть между 60 и 86400.
- `reply_markup` (*`telebot.types.InlineKeyboardMarkup`* or *`telebot.types.ReplyKeyboardMarkup`* or *`telebot.types.ReplyKeyboardRemove`* or *`telebot.types.ForceReply`*) – Дополнительные элементы интерфейса. Inline клавиатура, текстовая клавиатура, запрос на удаление текстовой клавиатуры или запрос на ответ от пользователя.
- `disable_notification` (`bool`) – Отправить сообщение, при получении которого пользователи получат уведомление без звука.
- `reply_to_message_id` (`int`) – deprecated.
- `allow_sending_without_reply` (`bool`) – deprecated.
- `timeout` (`int`) – Таймаут запроса в секундах.
- `horizontal_accuracy` (`float`) – Радиус погрешности местоположения, измеряется в метрах; 0-1500
- `heading` (`int`) – Для live местоположений, направление, в котором пользователь двигается, в градусах. Должно быть между 1 и 360, если указано.
- `proximity_alert_radius` (`int`) – Для live местоположений, максимальное расстояние для уведомлений о приближении другого участника чата, в метрах. Должно быть между 1 и 100000, если указано.
- `protect_content` (`bool`) – Запретить пересылку и сохранение содержимого сообщения
- `message_thread_id` (`int`) – id топика, в который нужно отправить сообщение
- `reply_parameters` (*`telebot.types.ReplyParameters`*) – Reply parameters.
- `business_connection_id` (`str`) – Identifier of a business connection, in which the message will be sent

Результат

В случае успеха возвращает отправленное сообщение (Message).

Тип результата

`telebot.types.Message`

`send_media_group`(*chat_id: int | str*, *media: List[*InputMediaAudio | InputMediaDocument | InputMediaPhoto | InputMediaVideo*]*, *disable_notification: bool | None = None*, *protect_content: bool | None = None*, *reply_to_message_id: int | None = None*, *timeout: int | None = None*, *allow_sending_without_reply: bool | None = None*, *message_thread_id: int | None = None*, *reply_parameters:* ReplyParameters | *None = None*, *business_connection_id: str | None = None*) → List[*Message*]

Используйте этот метод, чтобы отправить группу фото, видео, файлов или аудио как альбом. Файлы и аудио могут быть сгруппированы в альбом только с сообщениями того же типа. В случае успеха возвращается массив отправленных сообщений (Message).

Документация Telegram: https://core.telegram.org/bots/api#sendmediagroup

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `media` (`list` of *`types.InputMedia`*) – JSON-сериализованный массив, описывающий сообщения для отправки, должен включать от 2 до 10 элементов
- `disable_notification` (`bool`) – Отправить сообщение, при получении которого пользователя пользователи получат уведомление без звука.
- `protect_content` (`bool`) – Запретить пересылку и сохранение содержимого сообщения
- `reply_to_message_id` (`int`) – deprecated.
- `allow_sending_without_reply` (`bool`) – deprecated.
- `timeout` (`int`) – Таймаут запроса в секундах.
- `message_thread_id` (`int`) – id топика, в который будет отправлена группа медиа
- `reply_parameters` (*`telebot.types.ReplyParameters`*) – Reply parameters.
- `business_connection_id` (`str`) – Identifier of a business connection, in which the message will be sent

Результат

В случае успеха возвращается массив отправленных сообщений (Message).

Тип результата

List[*types.Message*]

`send_message(`*chat_id: int | str, text: str, parse_mode: str | None = None, entities: List[MessageEntity] | None = None, disable_web_page_preview: bool | None = None, disable_notification: bool | None = None, protect_content: bool | None = None, reply_to_message_id: int | None = None, allow_sending_without_reply: bool | None = None, reply_markup: InlineKeyboardMarkup | ReplyKeyboardMarkup | ReplyKeyboardRemove | ForceReply | None = None, timeout: int | None = None, message_thread_id: int | None = None, reply_parameters: ReplyParameters | None = None, link_preview_options: LinkPreviewOptions | None = None, business_connection_id: str | None = None*`)` → *Message*

Используйте этот метод, чтобы отправлять текстовые сообщения.

Предупреждение: Не отправляйте больше 4096 символов в одном сообщении, иначе вы рискуете получить ошибку HTTP 414. Если вам нужно отправить больше 4096 символов, используйте функцию *split_string* или *smart_split* из util.py.

Документация Telegram: https://core.telegram.org/bots/api#sendmessage

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `text` (`str`) – Текст сообщения для отправки

- parse_mode (str) – Режим форматирования в тексте сообщения.
- entities (Array of *telebot.types.MessageEntity*) – Список отформатированных частей в тексте сообщения, можно использовать вместо parse_mode
- disable_web_page_preview (bool) – deprecated.
- disable_notification (bool) – Отправить сообщение, при получении которого пользователи получат уведомление без звука.
- protect_content (bool) – Запретить пересылку и сохранение содержимого сообщения
- reply_to_message_id (int) – deprecated.
- allow_sending_without_reply (bool) – deprecated.
- reply_markup (*telebot.types.InlineKeyboardMarkup* or *telebot.types.ReplyKeyboardMarkup* or *telebot.types.ReplyKeyboardRemove* or *telebot.types.ForceReply*) – Дополнительные элементы интерфейса. Inline клавиатура, текстовая клавиатура, запрос на удаление текстовой клавиатуры или запрос на ответ от пользователя.
- timeout (int) – Таймаут запроса в секундах.
- message_thread_id (int) – id топика, в который нужно отправить сообщение
- reply_parameters (*telebot.types.ReplyParameters*) – Reply parameters.
- link_preview_options (*telebot.types.LinkPreviewOptions*) – Link preview options.
- business_connection_id (str) – Identifier of a business connection, in which the message will be sent

Результат

В случае успеха возвращает отправленное сообщение (Message).

Тип результата

telebot.types.Message

send_photo(*chat_id: int | str*, *photo: Any | str*, *caption: str | None = None*, *parse_mode: str | None = None*, *caption_entities: List[MessageEntity] | None = None*, *disable_notification: bool | None = None*, *protect_content: bool | None = None*, *reply_to_message_id: int | None = None*, *allow_sending_without_reply: bool | None = None*, *reply_markup: InlineKeyboardMarkup | ReplyKeyboardMarkup | ReplyKeyboardRemove | ForceReply | None = None*, *timeout: int | None = None*, *message_thread_id: int | None = None*, *has_spoiler: bool | None = None*, *reply_parameters: ReplyParameters | None = None*, *business_connection_id: str | None = None*) → *Message*

Используйте этот метод, чтобы отправить фото. В случае успеха возвращается отправленное сообщение (Message).

Документация Telegram: https://core.telegram.org/bots/api#sendphoto

Параметры

- chat_id (int or str) – Уникальный id чата или username канала (в формате @channelusername)
- photo (str or *telebot.types.InputFile*) – Фото для отправки. Передайте file_id (String), чтобы отправить фото, которое уже загружено на сервера Telegram (рекомендуется), передайте HTTP URL (String), чтобы отправить фото из интернета или загрузите новое с помощью multipart/form-data. Фото

должно весить не более 10 MB. Ширина и высота фото не должны суммарно превышать 10000. Отношение ширины и высоты должно быть не более 20.

- `caption` (`str`) – Подпись к фото (может быть использована при повторной отправке файла по file_id), 0-1024 символа после форматирования
- `parse_mode` (`str`) – Режим форматирования подписи к фото.
- `caption_entities` (`list of` *`telebot.types.MessageEntity`*) – Список отформатированных частей подписи в формате JSON, можно использовать вместо parse_mode
- `disable_notification` (`bool`) – Отправить сообщение, при получении которого пользователи получат уведомление без звука.
- `protect_content` (`bool`) – Запретить пересылку и сохранение содержимого сообщения
- `reply_to_message_id` (`int`) – deprecated.
- `allow_sending_without_reply` (`bool`) – deprecated.
- `reply_markup` (*`telebot.types.InlineKeyboardMarkup`* or *`telebot.types.ReplyKeyboardMarkup`* or *`telebot.types.ReplyKeyboardRemove`* or *`telebot.types.ForceReply`*) – Дополнительные элементы интерфейса. Inline клавиатура, текстовая клавиатура, запрос на удаление текстовой клавиатуры или запрос на ответ от пользователя.
- `timeout` (`int`) – Таймаут запроса в секундах.
- `message_thread_id` (`int`) – id топика, в который нужно отправить сообщение
- `has_spoiler` (`bool`) – Передайте True, если фото должно быть отправлено как спойлер
- `reply_parameters` (*`telebot.types.ReplyParameters`*) – Additional parameters for replies to messages
- `business_connection_id` (`str`) – Identifier of a business connection, in which the message will be sent

Результат

В случае успеха возвращает отправленное сообщение (Message).

Тип результата

`telebot.types.Message`

`send_poll(`*chat_id: int | str, question: str, options: List[str], is_anonymous: bool | None = None, type: str | None = None, allows_multiple_answers: bool | None = None, correct_option_id: int | None = None, explanation: str | None = None, explanation_parse_mode: str | None = None, open_period: int | None = None, close_date: int | datetime | None = None, is_closed: bool | None = None, disable_notification: bool | None = False, reply_to_message_id: int | None = None, reply_markup:* InlineKeyboardMarkup | ReplyKeyboardMarkup | ReplyKeyboardRemove | ForceReply *| None = None, allow_sending_without_reply: bool | None = None, timeout: int | None = None, explanation_entities: List[*MessageEntity*] | None = None, protect_content: bool | None = None, message_thread_id: int | None = None, reply_parameters:* ReplyParameters *| None = None, business_connection_id: str | None = None*`) →` *Message*

Используйте этот метод, чтобы отправить опрос. В случае успеха возвращается отправленное сообщение (Message).

Документация Telegram: https://core.telegram.org/bots/api#sendpoll

Параметры

- `chat_id` (`int` | `str`) – Уникальный id чата или username канала
- `question` (`str`) – Тема опроса, 1-300 символов
- `options` (`list` of `str`) – JSON-сериализованный список вариантов ответа, 2-10 строк по 1-100 символов
- `is_anonymous` (`bool`) – True, если опрос должен быть анонимным, по умолчанию True
- `type` (`str`) – Вид опроса, "quiz" или "regular", по умолчанию "regular"
- `allows_multiple_answers` (`bool`) – True, если опрос позволяет выбрать несколько вариантов ответа, игнорируется в опросах вида "quiz", по умолчанию False
- `correct_option_id` (`int`) – Индекс правильного варианта ответа, начиная с 0. Доступно только для опросов вида "quiz", по умолчанию None
- `explanation` (`str`) – Текст, который будет показан при выборе неправильно варианта ответа или нажатии на иконку лампочки в опросах вида "quiz", 0-200 символов и не более 2 строк после форматирования
- `explanation_parse_mode` (`str`) – Режим форматирования explanation. См. formatting options для получения подробностей.
- `open_period` (`int`) – Время в секундах, в течение которого опрос будет активен, 5-600. Нельзя использовать вместо с close_date.
- `close_date` (`int` | `datetime`) – Время (UNIX timestamp), когда опрос будет автоматически завершен.
- `is_closed` (`bool`) – Передайте True, если опрос должен быть завершен немедленно. Может быть полезно для предпросмотра опроса.
- `disable_notification` (`bool`) – Отправить сообщение, при получении которого пользователи получат уведомление без звука.
- `reply_to_message_id` (`int`) – deprecated. If the message is a reply, ID of the original message
- `allow_sending_without_reply` (`bool`) – deprecated. Pass True, if the message should be sent even if the specified replied-to message is not found
- `reply_markup` (`InlineKeyboardMarkup` | `ReplyKeyboardMarkup` | `ReplyKeyboardRemove` | `ForceReply`) – Дополнительные элементы интерфейса. Inline клавиатура, текстовая клавиатура, запрос на удаление текстовой клавиатуры или запрос на ответ от пользователя.
- `timeout` (`int`) – Тайм-аут в секундах, ожидание ответа от пользователя.
- `explanation_entities` (`list` of `MessageEntity`) – JSON-сериализованный список отформатированных частей explanation, можно использовать вместо parse_mode
- `protect_content` (`bool`) – Запретить пересылку и сохранение содержимого сообщения
- `message_thread_id` (`int`) – id топика, в который будет отправлен опрос
- `reply_parameters` (`ReplyParameters`) – reply parameters.

- business_connection_id (str) – Identifier of the business connection to use for the poll

Результат
В случае успеха возвращает отправленное сообщение (Message).

Тип результата
types.Message

send_sticker(*chat_id: int | str, sticker: Any | str, reply_to_message_id: int | None = None, reply_markup: InlineKeyboardMarkup | ReplyKeyboardMarkup | ReplyKeyboardRemove | ForceReply | None = None, disable_notification: bool | None = None, timeout: int | None = None, allow_sending_without_reply: bool | None = None, protect_content: bool | None = None, data: Any | str = None, message_thread_id: int | None = None, emoji: str | None = None, reply_parameters: ReplyParameters | None = None, business_connection_id: str | None = None*) → *Message*

Используйте этот метод, чтобы отправить статичный .WEBP, анимированный .TGS, или видео .WEBM стикер. В случае успеха возвращает отправленное сообщение (Message).

Документация Telegram: https://core.telegram.org/bots/api#sendsticker

Параметры

- chat_id (int or str) – Уникальный id чата или username канала (в формате @channelusername)
- sticker (str or *telebot.types.InputFile*) – Стикер для отправки. Передайте file_id (String), чтобы отправить файл, который уже загружен на сервера Telegram (рекомендуется), передайте HTTP URL (String), чтобы отправить .webp файл из интернета или загрузите новый с помощью multipart/form-data.
- reply_markup (*telebot.types.InlineKeyboardMarkup* or *telebot.types.ReplyKeyboardMarkup* or *telebot.types.ReplyKeyboardRemove* or *telebot.types.ForceReply*) – Дополнительные элементы интерфейса. Inline клавиатура, текстовая клавиатура, запрос на удаление текстовой клавиатуры или запрос на ответ от пользователя.
- disable_notification (bool) – отключить уведомление
- reply_to_message_id (int) – deprecated.
- allow_sending_without_reply (bool) – deprecated.
- timeout (int) – Таймаут запроса в секундах.
- protect_content (bool) – Запретить пересылку и сохранение содержимого сообщения
- data (str) – опечатка: не используйте
- message_thread_id (int) – Топик, в которой сообщение будет отправлено
- emoji (str) – Emoji associated with the sticker; only for just uploaded stickers
- reply_parameters (*telebot.types.ReplyParameters*) – Reply parameters.
- business_connection_id (str) – Identifier of a business connection, in which the message will be sent

Результат
В случае успеха возвращает отправленное сообщение (Message).

Тип результата
telebot.types.Message

`send_venue(`*chat_id: int | str, latitude: float | None, longitude: float | None, title: str, address: str, foursquare_id: str | None = None, foursquare_type: str | None = None, disable_notification: bool | None = None, reply_to_message_id: int | None = None, reply_markup:* InlineKeyboardMarkup | ReplyKeyboardMarkup | ReplyKeyboardRemove | ForceReply *| None = None, timeout: int | None = None, allow_sending_without_reply: bool | None = None, google_place_id: str | None = None, google_place_type: str | None = None, protect_content: bool | None = None, message_thread_id: int | None = None, reply_parameters:* ReplyParameters *| None = None, business_connection_id: str | None = None*`)` → *Message*

Используйте этот метод, чтобы отправить информацию о месте. В случае успеха возвращается отправленное сообщение (Message).

Документация Telegram: https://core.telegram.org/bots/api#sendvenue

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала
- `latitude` (`float`) – Широта
- `longitude` (`float`) – Долгота
- `title` (`str`) – Название места
- `address` (`str`) – Адрес места
- `foursquare_id` (`str`) – id места на Foursquare
- `foursquare_type` (`str`) – Тип места на Foursquare, если известен. (Например, "arts_entertainment/default", "arts_entertainment/aquarium" или "food/icecream".)
- `disable_notification` (`bool`) – Отправить сообщение, при получении которого пользователи получат уведомление без звука.
- `reply_to_message_id` (`int`) – deprecated.
- `allow_sending_without_reply` (`bool`) – deprecated.
- `reply_markup` (*telebot.types.InlineKeyboardMarkup* or *telebot.types.ReplyKeyboardMarkup* or *telebot.types.ReplyKeyboardRemove* or *telebot.types.ForceReply*) – Дополнительные элементы интерфейса. Inline клавиатура, текстовая клавиатура, запрос на удаление текстовой клавиатуры или запрос на ответ от пользователя.
- `timeout` (`int`) – Таймаут запроса в секундах.
- `google_place_id` (`str`) – id места на Google Places
- `google_place_type` (`str`) – Тип места на Google Places.
- `protect_content` (`bool`) – Запретить пересылку и сохранение содержимого сообщения
- `message_thread_id` (`int`) – id топика, на сообщение из которого нужно ответить
- `reply_parameters` (*telebot.types.ReplyParameters*) – Reply parameters.
- `business_connection_id` (`str`) – Identifier of a business connection

Результат
В случае успеха возвращает отправленное сообщение (Message).

Тип результата
telebot.types.Message

`send_video(`*chat_id: int | str, video: Any | str, duration: int | None = None, width: int | None = None, height: int | None = None, thumbnail: str | Any | None = None, caption: str | None = None, parse_mode: str | None = None, caption_entities: List[*MessageEntity*] | None = None, supports_streaming: bool | None = None, disable_notification: bool | None = None, protect_content: bool | None = None, reply_to_message_id: int | None = None, allow_sending_without_reply: bool | None = None, reply_markup:* InlineKeyboardMarkup | ReplyKeyboardMarkup | ReplyKeyboardRemove | ForceReply *| None = None, timeout: int | None = None, data: str | Any | None = None, message_thread_id: int | None = None, has_spoiler: bool | None = None, thumb: str | Any | None = None, reply_parameters:* ReplyParameters *| None = None, business_connection_id: str | None = None*`) → *Message*`

Используйте этот метод, чтобы отправить видео, клиенты (приложения) Telegram поддерживают mp4 видео (другие форматы могут быть отправлены как Document).

Документация Telegram: https://core.telegram.org/bots/api#sendvideo

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `video` (`str` or *telebot.types.InputFile*) – Видео для отправки. Передайте file_id (String), чтобы отправить видео, которое уже загружено на сервера Telegram или загрузите новое с помощью multipart/form-data.
- `duration` (`int`) – Длительность отправленного видео в секундах
- `width` (`int`) – Ширина видео
- `height` (`int`) – Высота видео
- `thumbnail` (`str` or *telebot.types.InputFile*) – Обложка отправленного файла; может быть проигнорирована, если генерация обложки поддерживается на стороне сервера. Обложка должна быть картинкой в формате JPEG и весить менее 200 kB. Ширина и высота обложки не должны превышать 320. Игнорируется, если файл не загружен с помощью multipart/form-data. Обложки не могут быть использованы повторно и могут быть загружены только как новый файл, так что вы можете передать "attach://<file_attach_name>" если обложка была загружена с помощью multipart/form-data под именем <file_attach_name>.
- `caption` (`str`) – Подпись к видео (может быть использована при повторной отправке файла по file_id), 0-1024 символа после форматирования
- `parse_mode` (`str`) – Режим форматирования подписи к видео
- `caption_entities` (`list` of *telebot.types.MessageEntity*) – Список отформатированных частей подписи, можно использовать вместо parse_mode
- `supports_streaming` (`bool`) – Передайте True, если загруженное видео подходит для стриминга
- `disable_notification` (`bool`) – Отправить сообщение, при получении которого пользователи получат уведомление без звука.

- `protect_content` (`bool`) – Запретить пересылку и сохранение содержимого сообщения
- `reply_to_message_id` (`int`) – deprecated.
- `allow_sending_without_reply` (`bool`) – deprecated.
- `reply_markup` (*telebot.types.InlineKeyboardMarkup* or *telebot.types.ReplyKeyboardMarkup* or *telebot.types.ReplyKeyboardRemove* or *telebot.types.ForceReply*) – Дополнительные элементы интерфейса. Inline клавиатура, текстовая клавиатура, запрос на удаление текстовой клавиатуры или запрос на ответ от пользователя.
- `timeout` (`int`) – Таймаут запроса в секундах.
- `data` (`str`) – опечатка: не используйте
- `message_thread_id` (`int`) – id топика, в который будет отправлено видео
- `has_spoiler` (`bool`) – Передайте True, если видео должно быть отправлено как спойлер
- `thumb` (`str` or *telebot.types.InputFile*) – deprecated.
- `reply_parameters` (*telebot.types.ReplyParameters*) – Reply parameters
- `business_connection_id` (`str`) – Identifier of a business connection

Результат

В случае успеха возвращает отправленное сообщение (Message).

Тип результата

telebot.types.Message

`send_video_note`(*chat_id: int | str*, *data: Any | str*, *duration: int | None = None*, *length: int | None = None*, *reply_to_message_id: int | None = None*, *reply_markup:* InlineKeyboardMarkup | ReplyKeyboardMarkup | ReplyKeyboardRemove | ForceReply | *None = None*, *disable_notification: bool | None = None*, *timeout: int | None = None*, *thumbnail: str | Any | None = None*, *allow_sending_without_reply: bool | None = None*, *protect_content: bool | None = None*, *message_thread_id: int | None = None*, *thumb: str | Any | None = None*, *reply_parameters:* ReplyParameters | *None = None*, *business_connection_id: str | None = None*) → *Message*

Начиная с версии v.4.0, клиенты(приложения) Telegram поддерживают скругленные квадратные MPEG4 видео длительностью до минуты. Используйте этот метод, чтобы отправить видео заметку (кружочек). В случае успеха возвращается отправленное сообщение (Message).

Документация Telegram: https://core.telegram.org/bots/api#sendvideonote

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `data` (`str` or *telebot.types.InputFile*) – Видео заметка для отправки. Передайте file_id (String), чтобы отправить видео заметку, которая уже загружена на сервера Telegram или загрузите новую с помощью multipart/form-data. На текущий момент, отправка видео заметок по URL не поддерживается
- `duration` (`int`) – Длительность отправленного видео в секундах
- `length` (`int`) – Ширина и высота видео (диаметр видео сообщения)

- reply_markup (*telebot.types.InlineKeyboardMarkup* or *telebot.types.ReplyKeyboardMarkup* or *telebot.types.ReplyKeyboardRemove* or *telebot.types.ForceReply*) – Дополнительные элементы интерфейса. Inline клавиатура, текстовая клавиатура, запрос на удаление текстовой клавиатуры или запрос на ответ от пользователя.
- disable_notification (bool) – Отправить сообщение, при получении которого пользователи получат уведомление без звука.
- reply_to_message_id (int) – deprecated.
- allow_sending_without_reply (bool) – deprecated.
- timeout (int) – Таймаут запроса в секундах.
- thumbnail (str or *telebot.types.InputFile*) – Обложка отправленного файла; может быть проигнорирована, если генерация обложки поддерживается на стороне сервера. Обложка должна быть картинкой в формате JPEG и весить менее 200 kB. Ширина и высота обложки не должны превышать 320. Игнорируется, если файл не загружен с помощью multipart/form-data. Обложки не могут быть использованы повторно и могут быть загружены только как новый файл, так что вы можете передать "attach://<file_attach_name>" если обложка была загружена с помощью multipart/form-data под именем <file_attach_name>.
- protect_content (bool) – Запретить пересылку и сохранение содержимого сообщения
- message_thread_id (int) – id топика, в который будет отправлена видео заметка
- thumb (str or *telebot.types.InputFile*) – deprecated.
- reply_parameters (*telebot.types.ReplyParameters*) – Reply parameters.
- business_connection_id (str) – Identifier of a business connection, in which the message will be sent

Результат

В случае успеха возвращает отправленное сообщение (Message).

Тип результата

telebot.types.Message

send_voice(*chat_id: int | str*, *voice: Any | str*, *caption: str | None = None*, *duration: int | None = None*, *reply_to_message_id: int | None = None*, *reply_markup: InlineKeyboardMarkup | ReplyKeyboardMarkup | ReplyKeyboardRemove | ForceReply | None = None*, *parse_mode: str | None = None*, *disable_notification: bool | None = None*, *timeout: int | None = None*, *caption_entities: List[MessageEntity] | None = None*, *allow_sending_without_reply: bool | None = None*, *protect_content: bool | None = None*, *message_thread_id: int | None = None*, *reply_parameters: ReplyParameters | None = None*, *business_connection_id: str | None = None*) → *Message*

Используйте этот метод, чтобы отправить голосовое сообщение. Ваше аудио должно быть в формате .OGG и закодировано с помощью OPUS (другие форматы можно отправить как Audio или Document). В случае успеха возвращается отправленное сообщение (Message). На текущий момент, боты могут отправлять голосовые сообщения весом до 50 MB, это ограничение может быть изменено в будущем.

Документация Telegram: https://core.telegram.org/bots/api#sendvoice

Параметры

- chat_id (int or str) – Уникальный id чата или username канала (в формате @channelusername)
- voice (str or *telebot.types.InputFile*) – Аудио для отправки. Передайте file_id (String), чтобы отправить аудио, которое уже загружено на сервера Telegram (рекомендуется), передайте HTTP URL (String), чтобы отправить аудио из интернета или загрузите новое с помощью multipart/form-data.
- caption (str) – Подпись к голосовому сообщению, 0-1024 символа после форматирования
- duration (int) – Длительность голосового сообщения в секундах
- reply_markup (*telebot.types.InlineKeyboardMarkup* or *telebot.types.ReplyKeyboardMarkup* or *telebot.types.ReplyKeyboardRemove* or *telebot.types.ForceReply*) – Дополнительные элементы интерфейса. Inline клавиатура, текстовая клавиатура, запрос на удаление текстовой клавиатуры или запрос на ответ от пользователя.
- parse_mode (str) – Режим форматирования подписи к голосовому сообщению. См. formatting options для получения подробностей.
- disable_notification (bool) – Отправить сообщение, при получении которого пользователи получат уведомление без звука.
- reply_to_message_id (int) – deprecated.
- allow_sending_without_reply (bool) – deprecated.
- timeout (int) – Таймаут запроса в секундах.
- caption_entities (list of *telebot.types.MessageEntity*) – Список отформатированных частей подписи в формате JSON, можно использовать вместо parse_mode
- protect_content (bool) – Запретить пересылку и сохранение содержимого сообщения
- message_thread_id (int) – id топика, в который нужно отправить сообщение
- reply_parameters (*telebot.types.ReplyParameters*) – Reply parameters.
- business_connection_id (str) – Identifier of a business connection, in which the message will be sent

Результат

В случае успеха возвращает отправленное сообщение (Message).

set_chat_administrator_custom_title(*chat_id: int / str*, *user_id: int*, *custom_title: str*) → bool

Используйте этот метод, чтобы задать кастомное звание администратора супергруппы, повышенного ботом. Возвращает True в случае успеха.

Документация Telegram: https://core.telegram.org/bots/api#setchatadministratorcustomtitle

Параметры

- chat_id (int or str) – Уникальный id чата или username супергруппы (в формате @supergroupusername)
- user_id (int) – Уникальный id сделавшего запрос пользователя
- custom_title (str) – Новое кастомное звание администратора; 0-16 символов, эмодзи не разрешены

Результат
True в случае успеха.

Тип результата
`bool`

`set_chat_description(chat_id: int | str, description: str | None = None)` → bool

Используйте этот метод, чтобы изменить описание супергруппы или канала. Бот должен быть администратором чата и иметь соответствующие права администратора.

Документация Telegram: https://core.telegram.org/bots/api#setchatdescription

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `description` (`str`) – Str: Новое описание чата, 0-255 символов

Результат
True в случае успеха.

Тип результата
`bool`

`set_chat_menu_button(chat_id: int | str = None, menu_button: MenuButton = None)` → bool

Используйте этот метод, чтобы изменить кнопку меню в приватном чате или кнопку меню по умолчанию. Возвращает True в случае успеха.

Документация Telegram: https://core.telegram.org/bots/api#setchatmenubutton

Параметры

- `chat_id` (`int` or `str`) – Уникальный id приватного чата. Если не указан, будет изменена кнопка меню по умолчанию.
- `menu_button` (*`telebot.types.MenuButton`*) – JSON-сериализованный объект новой кнопки меню. По умолчанию MenuButtonDefault

Результат
True в случае успеха.

Тип результата
`bool`

`set_chat_permissions(chat_id: int | str, permissions: ChatPermissions, use_independent_chat_permissions: bool | None = None)` → bool

Используйте этот метод, чтобы задать права по умолчанию для всех участников чата. Бот должен быть администратором группы или супергруппы и иметь права администратора can_restrict_members.

Документация Telegram: https://core.telegram.org/bots/api#setchatpermissions

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username супергруппы (в формате @supergroupusername)
- `permissions` (`telebot.types..ChatPermissions`) – Новые права по умолчанию
- `use_independent_chat_permissions` (`bool`) – Pass True if chat permissions are set independently. Otherwise, the can_send_other_messages and can_add_web_page_previews permissions will imply the can_send_messages,

can_send_audios, can_send_documents, can_send_photos, can_send_videos, can_send_video_notes, and can_send_voice_notes permissions; the can_send_polls permission will imply the can_send_messages permission.

Результат
True в случае успеха

Тип результата
`bool`

`set_chat_photo(chat_id: int | str, photo: Any)` → bool

Используйте этот метод, чтобы задать новую аватарку чата. В приватных чатах аватарки менять нельзя. Бот должен быть администратором чата и иметь соответствующие права администратора. Возвращает True в случае успеха. Примечание: В обычных группах (не супергруппы), этот метод будет работать только если настройка 'All Members Are Admins' отключена.

Документация Telegram: https://core.telegram.org/bots/api#setchatphoto

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `photo` (`typing.Union[file_like, str]`) – InputFile: Новая аватарка чата, загруженная с помощью multipart/form-data

Результат
True в случае успеха.

Тип результата
`bool`

`set_chat_sticker_set(chat_id: int | str, sticker_set_name: str)` → *StickerSet*

Используйте этот метод, чтобы задать стикерпак супергруппы. Бот должен быть администратором чата и иметь соответствующие права администратора. Используйте атрибут can_set_sticker_set, возвращаемые методом getChat, чтобы проверить, что бот может использовать этот метод. Возвращает True в случае успеха.

Документация Telegram: https://core.telegram.org/bots/api#setchatstickerset

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username супергруппы (в формате @supergroupusername)
- `sticker_set_name` (`str`) – Имя стикерпака для установки в качестве стикерпака группы

Результат
Объект StickerSet

Тип результата
`telebot.types.StickerSet`

`set_chat_title(chat_id: int | str, title: str)` → bool

Используйте этот метод, чтобы изменить название чата. В приватных чатах изменить название нельзя. Бот должен быть администратором чата и иметь соответствующие права админа. Возвращает True в случае успеха. Примечание: В обычных группах (не супергруппы), этот метод будет работать только если настройка 'All Members Are Admins' отключена.

Документация Telegram: https://core.telegram.org/bots/api#setchattitle

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `title` (`str`) – Новое название чата, 1-255 символов

Результат
True в случае успеха.

Тип результата
`bool`

`set_custom_emoji_sticker_set_thumbnail`(*name: str*, *custom_emoji_id: str | None = None*) → bool

Use this method to set the thumbnail of a custom emoji sticker set. Returns True on success.

Параметры

- `name` (`str`) – Имя стикерпака
- `custom_emoji_id` (`str`) – Custom emoji identifier of a sticker from the sticker set; pass an empty string to drop the thumbnail and use the first sticker as the thumbnail.

Результат
Возвращает True в случае успеха.

Тип результата
`bool`

`set_game_score`(*user_id: int | str*, *score: int*, *force: bool | None = None*, *chat_id: int | str | None = None*, *message_id: int | None = None*, *inline_message_id: str | None = None*, *disable_edit_message: bool | None = None*) → *Message* | bool

Задаёт количество очков пользователя в игре.

Документация Telegram: https://core.telegram.org/bots/api#setgamescore

Параметры

- `user_id` (`int` or `str`) – id пользователя
- `score` (`int`) – Количество очков, должно быть неотрицательным
- `force` (`bool`) – Передайте True, если количество очков могут быть уменьшено. Может быть полезно при исправлении ошибок или бане читеров
- `chat_id` (`int` or `str`) – Обязательный, если не указан inline_message_id. Уникальный id чата или username канала (в формате @channelusername)
- `message_id` (`int`) – Обязательный, если не указан inline_message_id. id отправленного сообщения
- `inline_message_id` (`str`) – Обязательный, если не указаны chat_id и message_id. id inline сообщения
- `disable_edit_message` (`bool`) – Передайте True, если сообщение с игрой должно быть автоматически отредактировано, чтобы отобразить новый результат

Результат
В случае успеха, если сообщение было отправлено ботом, возвращает измененное сообщение (Message), иначе возвращает True.

Тип результата
types.Message or bool

set_message_reaction(*chat_id: int | str*, *message_id: int*, *reaction: List[ReactionType] | None = None*, *is_big: bool | None = None*) → bool

Use this method to change the chosen reactions on a message. Service messages can't be reacted to. Automatically forwarded messages from a channel to its discussion group have the same available reactions as messages in the channel. Returns True on success.

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

Параметры

- chat_id (int or str) – Уникальный id чата или username супергруппы или канала (в формате @channelusername)
- message_id (int) – Identifier of the message to set reaction to
- reaction (list of *telebot.types.ReactionType*) – New list of reaction types to set on the message. Currently, as non-premium users, bots can set up to one reaction per message. A custom emoji reaction can be used if it is either already present on the message or explicitly allowed by chat administrators.
- is_big (bool) – Pass True to set the reaction with a big animation

Результат
bool

set_my_commands(*commands: List[BotCommand]*, *scope: BotCommandScope | None = None*, *language_code: str | None = None*) → bool

Используйте этот метод, чтобы изменить список команд бота.

Документация Telegram: https://core.telegram.org/bots/api#setmycommands

Параметры

- commands (list of *telebot.types.BotCommand*) – Список объектов BotCommand. Можно задать не более 100 команд.
- scope (*telebot.types.BotCommandScope*) – Область видимости команд. По умолчанию BotCommandScopeDefault.
- language_code (str) – Двухбуквенный языковой код в формате ISO 639-1. Если не задан, изменения коснутся команд для всех пользователей в заданном поле видимости, не имеющих команд на их языке

Результат
True в случае успеха.

Тип результата
bool

set_my_default_administrator_rights(*rights: ChatAdministratorRights = None*, *for_channels: bool | None = None*) → bool

Используйте этот метод, чтобы изменить права администратора по умолчанию, запрашиваемые при добавлении бота в группу или канал в качестве администратора. Эти права будут предложены пользователям, но пользователи могут изменить список перед добавлением бота. Возвращает True в случае успеха.

Документация Telegram: https://core.telegram.org/bots/api#setmydefaultadministratorrights

Параметры

- `rights` (*telebot.types.ChatAdministratorRights*) – JSON-сериалиованный объект, описывающий новые права администратора по умолчанию. Если не указан, права администратора по умолчанию будут сброшены.
- `for_channels` (`bool`) – Передайте True, чтобы изменить права администратора по умолчанию в каналах. Иначе, будут изменены права администратора по умолчанию для групп и супергрупп.

Результат
True в случае успеха.

Тип результата
`bool`

`set_my_description`(*description: str | None = None*, *language_code: str | None = None*)

Use this method to change the bot's description, which is shown in the chat with the bot if the chat is empty. Returns True on success.

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

Параметры

- `description` (`str`) – New bot description; 0-512 characters. Pass an empty string to remove the dedicated description for the given language.
- `language_code` (`str`) – A two-letter ISO 639-1 language code. If empty, the description will be applied to all users for whose language there is no dedicated description.

Результат
True в случае успеха.

`set_my_name`(*name: str | None = None*, *language_code: str | None = None*)

Use this method to change the bot's name. Returns True on success.

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

Параметры

- `name` (`str`) – Optional. New bot name; 0-64 characters. Pass an empty string to remove the dedicated name for the given language.
- `language_code` (`str`) – Optional. A two-letter ISO 639-1 language code. If empty, the name will be shown to all users for whose language there is no dedicated name.

Результат
True в случае успеха.

`set_my_short_description`(*short_description: str | None = None*, *language_code: str | None = None*)

Use this method to change the bot's short description, which is shown on the bot's profile page and is sent together with the link when users share the bot. Returns True on success.

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

Параметры

- `short_description` (`str`) – New short description for the bot; 0-120 characters. Pass an empty string to remove the dedicated short description for the given language.

- `language_code` (`str`) – A two-letter ISO 639-1 language code. If empty, the short description will be applied to all users for whose language there is no dedicated short description.

Результат
True в случае успеха.

`set_state`(*user_id: int*, *state: int | str | State*, *chat_id: int | None = None*) → None

Задаёт новое состояние (стейт) пользователя.

Примечание: Вы должны указать и user id и chat id, чтобы задать состояние (стейт) пользователя в чате. Иначе, если вы укажете только user_id, chat_id будет равен user_id, что означает смену состояния (стейта) пользователя в его приватном чате с ботом.

Параметры

- `user_id` (`int`) – id пользователя
- `state` (`int` or `str` or `telebot.types.State`) – новое состояние (стейт). может быть строкой, числом или `telebot.types.State`
- `chat_id` (`int`) – id чата

Результат
None

`set_sticker_emoji_list`(*sticker: str*, *emoji_list: List[str]*) → bool

Use this method to set the emoji list of a custom emoji sticker set. Returns True on success.

Параметры

- `sticker` (`str`) – Sticker identifier
- `emoji_list` (`list` of `str`) – List of emoji

Результат
Возвращает True в случае успеха.

Тип результата
`bool`

`set_sticker_keywords`(*sticker: str*, *keywords: List[str] = None*) → bool

Use this method to change search keywords assigned to a regular or custom emoji sticker. The sticker must belong to a sticker set created by the bot. Returns True on success.

Параметры

- `sticker` (`str`) – File identifier of the sticker.
- `keywords` (`list` of `str`) – A JSON-serialized list of 0-20 search keywords for the sticker with total length of up to 64 characters

Результат
В случае успеха возвращается True.

Тип результата
`bool`

`set_sticker_mask_position(sticker: str, mask_position: MaskPosition = None) → bool`

Use this method to change the mask position of a mask sticker. The sticker must belong to a sticker set that was created by the bot. Returns True on success.

Параметры

- `sticker` (`str`) – File identifier of the sticker.
- `mask_position` (*`telebot.types.MaskPosition`*) – A JSON-serialized object for position where the mask should be placed on faces.

Результат
Возвращает True в случае успеха.

Тип результата
`bool`

`set_sticker_position_in_set(sticker: str, position: int) → bool`

Используйте этот метод, чтобы передвинуть стикер в стикерпаке, созданном ботом, на заданную позицию. Возвращает True в случае успеха.

Документация Telegram: https://core.telegram.org/bots/api#setstickerpositioninset

Параметры

- `sticker` (`str`) – id файла стикера
- `position` (`int`) – Новая позиция стикера в стикерпаке, начиная с нуля

Результат
В случае успеха возвращается True.

Тип результата
`bool`

`set_sticker_set_thumb(**kwargs)`

`set_sticker_set_thumbnail(name: str, user_id: int, thumbnail: Any | str = None, format: str | None = None) → bool`

Используйте этот метод, чтобы задать обложку стикерпака. Анимированные обложки могут быть заданы только для анимированных стикерпаков. Возвращает True в случае успеха.

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

Параметры

- `name` (`str`) – Имя стикерпака
- `user_id` (`int`) – id пользователя
- `thumbnail` (`filelike object`) – A .WEBP or .PNG image with the thumbnail, must be up to 128 kilobytes in size and have a width and height of exactly 100px, or a .TGS animation with a thumbnail up to 32 kilobytes in size (see https://core.telegram.org/stickers#animated-sticker-requirements for animated sticker technical requirements), or a WEBM video with the thumbnail up to 32 kilobytes in size; see https://core.telegram.org/stickers#video-sticker-requirements for video sticker technical requirements. 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. More information on Sending Files ». Animated and video sticker set thumbnails can't be uploaded via HTTP URL. If omitted, then the thumbnail is dropped and the first sticker is used as the thumbnail.

- `format` (`str`) – Format of the thumbnail, must be one of "static" for a .WEBP or .PNG image, "animated" for a .TGS animation, or "video" for a WEBM video

Результат
В случае успеха возвращается True.

Тип результата
`bool`

`set_sticker_set_title(`*name: str*, *title: str*`)` → bool

Use this method to set the title of a created sticker set. Returns True on success.

Параметры

- `name` (`str`) – Имя стикерпака
- `title` (`str`) – New sticker set title

Результат
Возвращает True в случае успеха.

Тип результата
`bool`

`set_update_listener(`*listener: Callable*`)`

Задаёт функцию-листенер, которая будет вызвана при получении нового апдейта.

Параметры
`listener` (*`Callable`*) – Функция-листенер.

`set_webhook(`*url: str | None = None*, *certificate: str | Any | None = None*, *max_connections: int | None = None*, *allowed_updates: List[str] | None = None*, *ip_address: str | None = None*, *drop_pending_updates: bool | None = None*, *timeout: int | None = None*, *secret_token: str | None = None*`)` → bool

Используйте этот метод, чтобы задать URL и получать входящие апдейты с помощью вебхука. Как только у бота появляется апдейт, он будет отправлен с помощью HTTPS POST запроса на заданный URL, содержащего JSON-сериализованный Update. В случае неудачного запроса, отправка апдейта будет отменена после разумного числа попыток. Возвращает True в случае успеха.

Если вы хотите удостовериться, что вебхук был задан вами, вы можете задать секретный токен в параметре secret_token. Если указан, запрос с апдейтом будет содержать хедер "X-Telegram-Bot-Api-Secret-Token" с секретным токеном в качестве значения.

Документация Telegram: https://core.telegram.org/bots/api#setwebhook

Параметры

- `url` (`str`, optional) – HTTPS URL для отправки апдейтов. Используйте пустую строку, чтобы удалить вебхук, по умолчанию None
- `certificate` (`str`, optional) – Загрузите публичный ключ вашего SSL сертификата, чтобы корневой сертификат мог быть проверен, по умолчанию None
- `max_connections` (`int`, optional) – Максимально-допустимое количество одновременных HTTPS соединений для доставки апдейтов, 1-100. По умолчанию 40. Используйте меньшие значения для уменьшения нагрузки на ваш сервер и большие значения, чтобы увеличить пропускную способность вашего бота, по умолчанию None
- `allowed_updates` (`list`, optional) – Список видов апдейтов, которые вы хотите получать, в формате JSON. Например, укажите ["message",

"edited_channel_post", "callback_query"], чтобы получать апдейты только этих видов. Полный список доступных видов апдейтов - util.update_types. Укажите пустой список, чтобы получать все апдейты, кроме chat_member (по умолчанию). Если не задан, будет использована последняя настройка. Пожалуйста учтите, чтобы этот параметр не влияет на апдейты, отправленные до вызова setWebhooks, поэтому нежелательные апдейты могут быть получены в течение короткого периода времени. По умолчанию None

- `ip_address` (`str`, optional) – Фиксированный IP адрес, который будет использоваться для отправки запросов к вебхуку вместо IP адреса, полученного через DNS, по умолчанию None
- `drop_pending_updates` (`bool`, optional) – Передайте True, чтобы удалить все предшествующие запуску бота апдейты, по умолчанию None
- `timeout` (`int`, optional) – Тайм-аут запроса, по умолчанию None
- `secret_token` (`str`, optional) – Секретный токен для отправки в хедере "X-Telegram-Bot-Api-Secret-Token" в каждом запросе с апдейтом, 1-256 символов. Разрешены только символы A-Z, a-z, 0-9, _ и -. Хедер полезен для, того чтобы удостовериться, что запрос приходитс вебхука, установленного вами. По умолчанию None

Результат
True в случае успеха.

Тип результата
`bool` если запрос был успешным.

`setup_middleware(middleware: BaseMiddleware)`

Регистрирует класс-middleware

Параметры
`middleware` (*`telebot.handler_backends.BaseMiddleware`*) – Наследник класса *`telebot.handler_backends.BaseMiddleware`*

Результат
None

`shipping_query_handler(func, **kwargs)`

Обрабатывает shipping query. Только для инвойсов с гибкой ценой. В качестве параметра передаёт в декорируемую функцию объект *`telebot.types.ShippingQuery`*.

Параметры

- `func` (`function`) – Функция, используемая в качестве фильтра
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`stop_bot()`

Останавливает работу бота и закрывает рабочий пул.

`stop_message_live_location(chat_id: int | str | None = None, message_id: int | None = None, inline_message_id: str | None = None, reply_markup: InlineKeyboardMarkup | None = None, timeout: int | None = None) → Message`

Используйте этот метод, чтобы остановить обновление live местоположения до истечения

live_period. В случае успеха, если сообщение не является inline сообщением,возвращается измененное сообщение (Message), иначе возвращается True.

Документация Telegram: https://core.telegram.org/bots/api#stopmessagelivelocation

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `message_id` (`int`) – Обязательный, если не указан inline_message_id. id сообщения live местоположением, которое нужно остановить
- `inline_message_id` (`str`) – Обязательный, если не указаны chat_id и message_id. id inline сообщения с live местоположением, которое нужно остановить
- `reply_markup` (*`telebot.types.InlineKeyboardMarkup`* or *`telebot.types.ReplyKeyboardMarkup`* or *`telebot.types.ReplyKeyboardRemove`* or *`telebot.types.ForceReply`*) – JSON-сериализованный объект новой inline клавиатуры.
- `timeout` (`int`) – Таймаут запроса в секундах.

Результат

В случае успеха, если сообщение не является inline сообщением, возвращается измененное сообщение (Message), иначе возвращается True.

Тип результата

`telebot.types.Message` or bool

`stop_poll`(*chat_id: int | str*, *message_id: int*, *reply_markup:* InlineKeyboardMarkup | *None = None*) → *Poll*

Используйте этот метод, чтобы завершить опрос, отправленный ботом. В случае успеха возвращается завершенный опрос (Poll).

Документация Telegram: https://core.telegram.org/bots/api#stoppoll

Параметры

- `chat_id` (`int` | `str`) – Уникальный id чата или username канала
- `message_id` (`int`) – id сообщения с опросом
- `reply_markup` (`InlineKeyboardMarkup`) – JSON-сериализованный объект новой inline клавиатуры.

Результат

В случае успеха возвращается завершенный опрос (Poll).

Тип результата

`types.Poll`

`stop_polling()`

Останавливает поллинг.

Не принимает никаких аргументов.

`unban_chat_member`(*chat_id: int | str*, *user_id: int*, *only_if_banned: bool | None = False*) → bool

Используйте этот метод, чтобы разбанить ранее кикнутого пользователя в супергруппе или канале. Пользовать не вернется в группу или канал автоматически, но сможет присоединиться с помощью ссылки и т.д. Бот должен быть администратором. По умолчанию, этот метод гарантирует, что после вызова, пользователь не является участником чата, но может

присоединиться. Поэтому если пользовать является участником чата, он будет кикнут, но не забанен. Если вы хотите изменить это поведение, используйте параметр only_if_banned.

Документация Telegram: https://core.telegram.org/bots/api#unbanchatmember

Параметры

- `chat_id` (`int` or `str`) – Уникальный id группы или username супергруппы или канала (в формате @username)
- `user_id` (`int`) – Уникальный id сделавшего запрос пользователя
- `only_if_banned` (`bool`) – Ничего не делать, если пользователь не забанен

Результат
True в случае успеха

Тип результата
`bool`

`unban_chat_sender_chat(`*chat_id: int | str*, *sender_chat_id: int | str*`)` → bool

Используйте этот метод, чтобы разбанить ране забаненный канал в супергруппе или канала. Бот должен быть администратором и иметь соответствующие права администратора. Возвращает True в случае успеха.

Документация Telegram: https://core.telegram.org/bots/api#unbanchatsenderchat

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `sender_chat_id` (`int` or `str`) – Уникальный id чата.

Результат
True в случае успеха.

Тип результата
`bool`

`unhide_general_forum_topic(`*chat_id: int | str*`)` → bool

Используйте этот метод, чтобы сделать топик „General“ видимым в супергруппе с топиками. Бот должен быть администратором чата и иметь права администратора can_manage_topics. Возвращает True в случае успеха.

Документация Telegram: https://core.telegram.org/bots/api#unhidegeneralforumtopic

Параметры
`chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)

`unpin_all_chat_messages(`*chat_id: int | str*`)` → bool

Используйте этот метод, что открепить все закрепленные сообщения в супергруппе. Бот должен быть администратором чата и иметь соответствующие права администратора. Возвращает True в случае успеха.

Документация Telegram: https://core.telegram.org/bots/api#unpinallchatmessages

Параметры
`chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)

Результат
True в случае успеха.

Тип результата
`bool`

`unpin_all_forum_topic_messages(chat_id: str | int, message_thread_id: int) → bool`

Используйте этот метод, что открепить все закрепленные сообщения в топике. Бот должен быть администратором чата и иметь права администратора can_pin_messages в супергруппе. Возвращает True в случае успеха.

Документация Telegram: https://core.telegram.org/bots/api#unpinallforumtopicmessages

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `message_thread_id` (`int`) – id топика

Результат
В случае успеха возвращается True.

Тип результата
`bool`

`unpin_all_general_forum_topic_messages(chat_id: int | str) → bool`

Use this method to clear the list of pinned messages in a General 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#unpinAllGeneralForumTopicMessages

Параметры
`chat_id` (`int` | `str`) – Unique identifier for the target chat or username of chat

Результат
В случае успеха возвращается True.

Тип результата
`bool`

`unpin_chat_message(chat_id: int | str, message_id: int | None = None) → bool`

Используйте этот метод, что открепить закрепленное сообщение в супергруппе. Бот должен быть администратором чата и иметь соответствующие права администратора. Возвращает True в случае успеха.

Документация Telegram: https://core.telegram.org/bots/api#unpinchatmessage

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `message_id` (`int`) – Int: id сообщения, которое нужно открепить

Результат
True в случае успеха.

Тип результата
`bool`

`upload_sticker_file(user_id: int, png_sticker: Any | str = None, sticker: InputFile | None = None, sticker_format: str | None = None) → File`

Используйте этот метод, чтобы загрузить .png стикер, чтобы позже использовать в методах

createNewStickerSet и addStickerToSet (может быть использован несколько раз). Возвращает загруженный файл (File) в случае успеха.

Документация Telegram: https://core.telegram.org/bots/api#uploadstickerfile

Параметры

- `user_id` (`int`) – id пользователя, создавшего стикерпак
- `png_sticker` (`filelike object`) – DEPRECATED: 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.
- `sticker` (*telebot.types.InputFile*) – A file with the sticker in .WEBP, .PNG, .TGS, or .WEBM format. See https://core.telegram.org/stickers for technical requirements. More information on Sending Files »
- `sticker_format` (`str`) – One of «static», «animated», «video».

Результат
В случае успеха возвращается отправленный файл.

Тип результата
telebot.types.File

`property user: User`

Объект User, описывающий бота. Эквивалент bot.get_me(), но результат кэшируется, поэтому нужен всего один запрос к API.

Результат
Информация о боте

Тип результата
telebot.types.User

Файл custom_filters

`class telebot.custom_filters.AdvancedCustomFilter`

Базовые классы: `ABC`

Базовый класс Advanced Custom Filter. Создайте класс наследник с методом check(). Принимает два параметра, возвращает bool: True - фильтр пройден, False - фильтр не пройден. message: класс Message text: значение фильтра, полученное в хендлере

Классы наследники должны иметь статический атрибут (property) .key

Список 9: Пример создания 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)`

Выполнить проверку.

`key: str = None`

class telebot.custom_filters.ChatFilter

Базовые классы: *AdvancedCustomFilter*

Проверяет, является ли chat_id заданным.

Список 10: Пример использования этого фильтра:

```
@bot.message_handler(chat_id=[99999])
# your function
```

key: str = 'chat_id'

class telebot.custom_filters.ForwardFilter

Базовые классы: *SimpleCustomFilter*

Проверяет, является ли сообщение пересланным из канала или группы.

Список 11: Пример использования этого фильтра:

```
@bot.message_handler(is_forwarded=True)
# your function
```

key: str = 'is_forwarded'

class telebot.custom_filters.IsAdminFilter(*bot*)

Базовые классы: *SimpleCustomFilter*

Проверяет, является ли пользователь администратором / владельцем чата.

Список 12: Пример использования этого фильтра:

```
@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*

Фильтр для проверки, состоит ли строка только из цифр.

Список 13: Пример использования этого фильтра:

```
@bot.message_handler(is_digit=True)
# your function
```

key: str = 'is_digit'

class telebot.custom_filters.IsReplyFilter

Базовые классы: *SimpleCustomFilter*

Проверяет, является ли сообщение ответом (reply).

Список 14: Пример использования этого фильтра:

```
@bot.message_handler(is_reply=True)
# your function
```

key: str = 'is_reply'

class telebot.custom_filters.LanguageFilter

Базовые классы: *AdvancedCustomFilter*

Проверяет language_code пользователя.

Список 15: Пример использования этого фильтра:

```
@bot.message_handler(language_code=['ru'])
# your function
```

key: str = 'language_code'

class telebot.custom_filters.SimpleCustomFilter

Базовые классы: ABC

Базовый класс Simple Custom Filter. Создайте класс наследник с методом check(). Принимает только сообщение, возвращает bool, который сравнивается с заданным в хендлере.

Классы наследники должны иметь статический атрибут (property) .key

Список 16: Пример создания simple custom filter.

```
class ForwardFilter(SimpleCustomFilter):
    # Check whether message was forwarded from channel or group.
    key = 'is_forwarded'

    def check(self, message):
        return message.forward_date is not None
```

check(*message*)

Выполнить проверку.

key: str = None

class telebot.custom_filters.StateFilter(*bot*)

Базовые классы: *AdvancedCustomFilter*

Фильтр для проверки состояния (стейта).

Список 17: Пример использования этого фильтра:

```
@bot.message_handler(state=1)
# your function
```

key: str = 'state'

class telebot.custom_filters.TextContainsFilter

Базовые классы: *AdvancedCustomFilter*

Фильтр для проверки текста сообщения. key: text

Список 18: Пример использования этого фильтра:

```
# 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: str | None = None, contains: list | tuple | None = None, starts_with: str | list | tuple | None = None, ends_with: str | list | tuple | None = None, ignore_case: bool = False*)

Базовые классы: object

Advanced текстовый фильтр для проверки (types.Message, types.CallbackQuery, types.InlineQuery, types.Poll)

пример использования в examples/custom_filters/advanced_text_filter.py

Параметры

- equals (str) – строка, True если текст объекта идентичен заданной строке
- contains (*list[str] or tuple[str]*) – list[str] или tuple[str], True если хотя бы один из элементов есть в тексте
- starts_with (str) – string, True если текст объекта начинается с заданной строки
- ends_with (str) – string, True если текст объекта начинается с заданной строки
- ignore_case (bool) – bool (по умолчанию False), независимый от регистра

Исключение

ValueError – если было задано некорректное значение параметра

Результат

None

class telebot.custom_filters.TextMatchFilter

Базовые классы: *AdvancedCustomFilter*

Фильтр для проверки текста сообщения.

Список 19: Пример использования этого фильтра:

```
@bot.message_handler(text=['account'])
# your function
```

key: str = 'text'

class telebot.custom_filters.TextStartsFilter

Базовые классы: *AdvancedCustomFilter*

Фильтр для проверки, начинается ли сообщение с заданного текста.

Список 20: Пример использования этого фильтра:

```
# Will work if message.text starts with 'sir'.
@bot.message_handler(text_startswith='sir')
# your function
```

key: str = 'text_startswith'

Файл handler_backends

class telebot.handler_backends.BaseMiddleware

Базовые классы: object

Базовый класс для middleware. Ваши middleware должны быть унаследованы от этого класса.

Задайте update_sensitive=True если хотите получать разные апдейты в разных функциях. Например, если вы хотите обрабатывать pre_process для апдейтов вида message, вам нужно будет создать функцию pre_process_message и т.д. Аналогично для post_process.

Примечание: Если вы хотите использовать middleware, вам нужно задать use_class_middlewares=True в экземпляре класса TeleBot.

Список 21: Пример класса middleware.

```
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):
        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

Класс для отмены апдейтов. Просто верните экземпляр этого класса в middleware, чтобы пропустить апдейт. Апдейт пропустит хендлер и исполнение post_process в middleware.

class telebot.handler_backends.ContinueHandling

Базовые классы: object

Класс для продолжения обработки апдейта в хендлерах. Просто верните экземпляр этого класса в хендлерах, чтобы продолжить обработку.

Список 22: Пример использования 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

Класс для пропуска хендлеров. Просто верните экземпляр этого класса в middleware, чтобы пропустить хендлер. Апдейт попадёт в post_process, но пропустит исполнение хендлера.

class telebot.handler_backends.State

Базовые классы: object

Класс, представляющий состояние (стейт).

```
class MyStates(StatesGroup):
    my_state = State() # returns my_state:State string.
```

class telebot.handler_backends.StatesGroup

Базовые классы: object

Класс, представляющий похожие состояния (стейты).

```
class MyStates(StatesGroup):
    my_state = State() # returns my_state:State string.
```

classmethod state_list()

Расширения

1.3.5 AsyncTeleBot

Методы класса AsyncTeleBot

class telebot.async_telebot.AsyncTeleBot(*token: str, parse_mode: str | None = None, offset: int | None = None, exception_handler: ~telebot.async_telebot.ExceptionHandler | None = None, state_storage: ~telebot.asyncio_storage.base_storage.StateStorageBase | None = <telebot.asyncio_storage.memory_storage.StateMemoryStorage object>, disable_web_page_preview: bool | None = None, disable_notification: bool | None = None, protect_content: bool | None = None, allow_sending_without_reply: bool | None = None, colorful_logs: bool | None = False*)

Базовые классы: `object`

Это основной класс для асинхронного бота.

Позволяет добавить хендлеры для различных апдейтов.

Использование:

Список 23: Использование асинхронной реализации TeleBot-а.

```
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.
```

Больше примеров в папке examples/ : https://github.com/eternnoir/pyTelegramBotAPI/tree/master/examples

Примечание: Установите пакет coloredlogs для использования colorful_los=True

Параметры

- `token` (`str`) – Токен бота, нужно получить от @BotFather
- `parse_mode` (`str`, optional) – Глобальный parse_mode, по умолчанию None
- `offset` (`int`, optional) – Смещение, используемое в get_updates, по умолчанию None
- `exception_handler` (*Optional [ExceptionHandler], optional*) – Класс для обработки исключений, по умолчанию None
- `state_storage` (`telebot.asyncio_storage.StateMemoryStorage`, optional) – Хранилище состояний (стейтов), по умолчанию StateMemoryStorage()
- `disable_web_page_preview` (`bool`, optional) – Глобальное значение disable_web_page_preview, по умолчанию None
- `disable_notification` (`bool`, optional) – Глобальное значение disable_notification, по умолчанию None
- `protect_content` (`bool`, optional) – Глобальное значение protect_content, по умолчанию None
- `allow_sending_without_reply` (`bool`, optional) – Deprecated - Use reply_parameters instead. Default value for allow_sending_without_reply, defaults to None
- `colorful_logs` (`bool`, optional) – Использовать разноцветные логи

`add_custom_filter(`*custom_filter:* SimpleCustomFilter / AdvancedCustomFilter`)`

Создать кастомный фильтр.

Список 24: Пример проверки текста сообщения

```
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*) – Класс с методом check(message)

Результат

None

`async add_data`(*user_id: int*, *chat_id: int | None = None*, ***kwargs*)

Добавить данные в состояние (стейт).

Параметры

- `user_id` (`int`) – id пользователя
- `chat_id` (`int`) – id чата
- `kwargs` – Данные для добавления

Результат

None

`async add_sticker_to_set`(*user_id: int*, *name: str*, *emojis: List[str] | str = None*, *png_sticker: str | Any | None = None*, *tgs_sticker: str | Any | None = None*, *webm_sticker: str | Any | None = None*, *mask_position:* MaskPosition *| None = None*, *sticker:* InputSticker *| None = None*) → bool

Use this method to add a new sticker to a set created by the bot. The format of the added sticker must match the format of the other stickers in the set. Emoji sticker sets can have up to 200 stickers. Animated and video sticker sets can have up to 50 stickers. Static sticker sets can have up to 120 stickers. Returns True on success.

Примечание: **_sticker, mask_position, emojis parameters are deprecated, use stickers instead

---

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

Параметры

- `user_id` (`int`) – id пользователя, создавшего стикерпак
- `name` (`str`) – Имя стикерпака
- `emojis` (`str`) – Один или несколько эмодзи, относящихся к стикеру
- `png_sticker` (`str` or `filelike object`) – Изображение стикера в формате PNG, весом не более 512 килобайт, размеры не должны превышать 512px, либо ширина, либо высота должны быть ровно 512px. Передайте file_id в формате str, чтобы отправить уже загруженный на сервера Telegram файл, передайте HTTP URL в формате str, чтобы Telegram скачал файл из интернета, или загрузите новый файл с помощью multipart/form-data.

- `tgs_sticker` (`str` or `filelike object`) – Анимированный стикер в формате TGS, загруженный с помощью multipart/form-data.
- `webm_sticker` (`str` or `filelike object`) – Анимированный стикер в формате WebM, загруженный с помощью multipart/form-data.
- `mask_position` (*`telebot.types.MaskPosition`*) – Позиция для размещения маски на лицах в формате JSON
- `sticker` (*`telebot.types.InputSticker`*) – A JSON-serialized object for sticker to be added to the sticker set

Результат
В случае успеха возвращается True.

Тип результата
`bool`

`async answer_callback_query(`*callback_query_id: int*, *text: str | None = None*, *show_alert: bool | None = None*, *url: str | None = None*, *cache_time: int | None = None*`)` → bool

Используйте этот метод для отправки ответов на callback запросы, отправленные с помощью inline кнопок. Ответ будет показан пользователю как уведомление поверх чата или pop-up предупреждение.

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

Параметры

- `callback_query_id` (`int`) – Уникальный id запроса для ответа
- `text` (`str`) – Текст уведомления. если не задан, то уведомление не будет показано, 0-200 символов
- `show_alert` (`bool`) – Если True, вместо уведомления поверх чата будет показано pop-up предупреждение, по умолчанию False.
- `url` (`str`) – URL, который будет открыт пользовательским клиентом. Если вы создали игру и приняли условия через @BotFather, задайте URL, открывающий вашу игру - учитывайте, что это сработает только если запрос был отправлен с помощью callback_game кнопки.
- `cache_time` – Максимальная длительность хранения ответа на callback запрос пользовательским клиентом в секундах. Приложения Telegram поддерживают хранение ответов начиная с версии 3.14, по умолчанию 0.

Результат
В случае успеха возвращается True.

Тип результата
`bool`

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

Используйте этот метод для отправки ответов на inline запрос. В случае успеха возвращается True. Разрешено отправить не более 50 результатов на один запрос.

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

Параметры

- `inline_query_id` (`str`) – Уникальный id запроса для ответа
- `results` (`list` of `types.InlineQueryResult`) – Массив результатов для ответа на inline запрос
- `cache_time` (`int`) – Максимальная длительность хранения результатов inline запроса на сервере в секундах.
- `is_personal` (`bool`) – Передайте True, если результаты должны быть сохранены на сервере только для пользователя, отправившего запрос.
- `next_offset` (`str`) – Передайте смещение, которое клиент должен отправить в следующем запросе с таким же текстом, чтобы получить новые результаты.
- `switch_pm_parameter` (`str`) – Параметр для команды /start, отправляемой боту, когда пользователь нажимает кнопку переключения. 1-64 символа, разрешены только A-Z, a-z, 0-9, _ и -. Пример: Inline бот, который отправляет видео с YouTube может попросить пользователя подключить бота к его YouTube аккаунту, чтобы поиск соответствовал предпочтениям пользователя. Чтобы это сделать, бот отправляет пользователю кнопку „Подключить YouTube аккаунт“ над результатами, или даже до их показа. Пользователь нажимает на кнопку, автоматически переходит в приватный чат с ботом и в это время передаёт стартовый параметр, по которому бот возвращает ссылку для авторизации (OAuth). Как только авторизация пройдена, бот может предложить switch_inline кнопку, чтобы пользователь мог легко вернуться в чат, где он хотел использовать возможности inline бота.
- `switch_pm_text` (`str`) – Параметр для передачи боту вместе с сообщением /start, отправленному при нажатии кнопки переключения
- `button` (`types.InlineQueryResultsButton`) – A JSON-serialized object describing a button to be shown above inline query results

Результат

В случае успеха возвращается True.

Тип результата

`bool`

`async answer_pre_checkout_query(`*pre_checkout_query_id: int*, *ok: bool*, *error_message: str | None = None*`)` → bool

Как только пользователь подтвердил детали оплаты и доставки, Bot API отправляет финальное подтверждение в виде апдейта с полем pre_checkout_query. Используйте этот метод для ответа на такие pre-checkout запросы. В случае успеха возвращается True.

Примечание: Bot API должно получить ответ в течение 10 секунд после отправки pre-checkout запроса.

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

Параметры

- `pre_checkout_query_id` (`int`) – Уникальный id запроса для ответа
- `ok` (`bool`) – Задайте True если всё правильно (выбранные товары доступны и т.д.) и бот готов обработать заказ. Задайте False если есть какие-то проблемы.

- `error_message` (`str`) – Обязательный в случае, когда ok - False. Сообщение об ошибке, которое может прочитать человек, объясняющее причину, по которой бот не может обработать заказ (например «Извините, кто-то только что купил последнюю из наших прекрасных черных футболок с коротким рукавом пока вы заполняли детали оплаты. Пожалуйста выберите другой цвет или фасон!»). Telegram покажет это сообщение пользователю.

Результат
В случае успеха возвращается True.

Тип результата
`bool`

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

Запрашивает ответ на вопрос о доставке.

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

Параметры

- `shipping_query_id` (`str`) – Уникальный id запроса для ответа
- `ok` (`bool`) – Задайте True если доставка по выбранному адресу возможна и False, если есть какие-то проблемы (например, доставка по выбранному адресу не осуществляется)
- `shipping_options` (`list of ShippingOption`) – Обязательный в случае, когда ok - True. Массив вариантов доставки в формате JSON.
- `error_message` (`str`) – Обязательный в случае, когда ok - False. Сообщение об ошибке, которое может прочитать человек, объясняющее причину, по которой невозможно завершить заказ (например «Извините, доставка по запрошенному адресу недоступна»). Telegram покажет это сообщение пользователю.

Результат
В случае успеха возвращается True.

Тип результата
`bool`

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

Используйте этот метод, чтобы задать результат взаимодействия с Web App и отправить соответствующее сообщение от лица пользователя в чат, из которого пришел запрос. В случае успеха возвращается объект SentWebAppMessage.

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

Параметры

- `web_app_query_id` (`str`) – Уникальный id запроса для ответа
- `result` (*`telebot.types.InlineQueryResultBase`*) – Объект в формате JSON, описывающий сообщение, которое нужно отправить

Результат
В случае успеха возвращается объект SentWebAppMessage.

Тип результата
`telebot.types.SentWebAppMessage`

`async approve_chat_join_request(chat_id: str | int, user_id: int | str)` → bool

Используйте этот метод, чтобы одобрить запрос на вступление в чат. Бот должен быть администратором чата и иметь права администратора can_invite_users. Возвращает True в случае успеха.

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

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username супергруппы (в формате @supergroupusername)
- `user_id` (`int` or `str`) – Уникальный id сделавшего запрос пользователя

Результат
True в случае успеха.

Тип результата
`bool`

`async ban_chat_member(chat_id: int | str, user_id: int, until_date: int | datetime | None = None, revoke_messages: bool | None = None)` → bool

Используйте этот метод, чтобы заблокировать пользователя в группе, супергруппе или канале. В случае супергрупп и каналов, пользователь не сможет вернуться в чат самостоятельно, используя ссылки с приглашением и т.д., пока не будет разблокирован. Возвращает True в случае успеха.

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

Параметры

- `chat_id` (`int` or `str`) – Уникальный id группы или username супергруппы или канала (в формате @channelusername)
- `user_id` (`int`) – Уникальный id сделавшего запрос пользователя
- `until_date` (`int` or `datetime`) – Дата, когда пользователь будет разблокирован, в формате UNIX time. Если пользователь заблокирован больше чем на 366 дней или меньше чем на 30 секунд, то он будет заблокирован до ручной разблокировки
- `revoke_messages` (`bool`) – Bool: Передайте True, чтобы удалить все сообщения пользователя из чата. Если False, пользователю будут доступны все сообщения в группе, отправленные до его блокировки. Всегда True для супергрупп и каналов.

Результат
Возвращает True в случае успеха.

Тип результата
`bool`

`async ban_chat_sender_chat(chat_id: int | str, sender_chat_id: int | str)` → bool

Используйте этот метод, чтобы заблокировать канал в супергруппе или канале. Владелец канала не сможет отправлять сообщения и участвовать в прямых эфирах от лица канала, пока канал не будет разблокирован. Бот должен быть администратором супергруппы или канала и иметь соответствующие права администратора.Возвращает True в случае успеха.

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

Параметры

- chat_id (int or str) – Уникальный id чата или username канала (в формате @channelusername)
- sender_chat_id (int or str) – Уникальный id канала для блокировки

Результат
True в случае успеха.

Тип результата
bool

business_connection_handler(*func=None*, ***kwargs*)

Handles new incoming business connection state.

Параметры

- func (function) – Функция, используемая в качестве фильтра
- kwargs – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

business_message_handler(*commands: List[str] | None = None*, *regexp: str | None = None*, *func: Callable | None = None*, *content_types: List[str] | None = None*, ***kwargs*)

Handles New incoming message of any kind(for business accounts, see bot api 7.2 for more) - 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.

Пример:

Список 25: Usage of business_message_handler

```
bot = TeleBot('TOKEN')

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

# Handle all sent documents of type 'text/plain'.
@bot.business_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.business_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) – Необязательный список строк - команд для обработки.

- `regexp` (`str`) – Необязательное регулярное выражение.
- `func` (`lambda`) – Необязательная lambda функция. Получает сообщение (объект Message) в качестве первого параметра. Функция должна вернуть True если хендлер должен обработать сообщение.
- `content_types` (`list` of `str`) – Обрабатываемые виды контента. Обязан быть списком. По умолчанию [„text“]
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
декорируемая функция

`callback_query_handler(func, **kwargs)`

Обрабатывает новый callback запрос. В качестве параметра передаёт в декорируемую функцию объект *telebot.types.CallbackQuery*.

Параметры

- `func` (`function`) – Функция, используемая в качестве фильтра
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

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

Обрабатывает новый пост любого типа в канале - текст, фото, стикер и т.д. В качестве параметра передаёт в декорируемую функцию объект *telebot.types.Message*.

Параметры

- `commands` (`list` of `str`) – Необязательный список строк - команд для обработки.
- `regexp` (`str`) – Необязательное регулярное выражение.
- `func` (`function`) – Функция, используемая в качестве фильтра
- `content_types` (`list` of `str`) – Обрабатываемые виды контента. Обязан быть списком. По умолчанию [„text“]
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

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

Handles new incoming chat boost state. it passes *telebot.types.ChatBoostUpdated* object.

Параметры

- `func` (`function`) – Функция, используемая в качестве фильтра
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

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

Обрабатывает запрос на вступление в чат. Бот должен иметь права администратора can_invite_users в чате, чтобы получать такие апдейты. В качестве параметра передаёт в декорируемую функцию объект *telebot.types.ChatJoinRequest*.

Параметры

- `func` (`function`) – Функция, используемая в качестве фильтра
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

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

Обрабатывает изменение статуса пользователя в чате. Бот должен быть администратором чата и явно указать "chat_member" в allowed_updates, чтобы получать такие апдейты. В качестве параметра передаёт в декорируемую функцию объект *telebot.types.ChatMemberUpdated*.

Параметры

- `func` (`function`) – Функция, используемая в качестве фильтра
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`chosen_inline_handler(func, **kwargs)`

Обрабатывает результат inline запроса, который был выбран пользователем и отправлен собеседнику в чате. Пожалуйста ознакомьтесь с документацией по сбору фидбека для получения таких апдейтов вашим ботом. В качестве параметра передаёт в декорируемую функцию объект *telebot.types.ChosenInlineResult*.

Параметры

- `func` (`function`) – Функция, используемая в качестве фильтра
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`async close()` → bool

Используйте этот метод чтобы закрыть инстанс бота прежде чем перемещать его с одного локального сервера на другой. Вы должны удалить вебхук перед вызовом этого метода, чтобы убедиться. что бот не будет запущен повторно после перезапуска сервера. Метод будет возвращать ошибку 429 в течение 10 минут после запуска бота. Возвращает True в случае успеха.

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

Результат
`bool`

`async close_forum_topic(chat_id: str | int, message_thread_id: int)` → bool

Используйте этот метод, чтобы закрыть открытый топик в чате супергруппы. Бот должен быть администратором чата и иметь права администратора can_manage_topics, за исключением случаев, когда бот является создателем топика. Возвращает True в случае успеха.

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

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)

- `message_thread_id` (`int`) – id топика для закрытия

Результат
В случае успеха возвращается True.

Тип результата
`bool`

`async close_general_forum_topic(chat_id: int | str)` → bool

Используйте этот метод, чтобы закрыть открытый топик в чате супергруппы. Бот должен быть администратором чата и иметь права администратора can_manage_topics, за исключением случаев, когда бот является создателем топика. Возвращает True в случае успеха.

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

Параметры
`chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)

`async close_session()`

Закрывает текущую aiohttp сессию. Используйте эту функцию для завершения работы поллинга или вебхука.

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

Используйте этот метод для копирования любых сообщений.

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

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `from_chat_id` (`int` or `str`) – Уникальный id чата, в который было отправлено исходное сообщение (или username канала в формате @channelusername)
- `message_id` (`int`) – id сообщения в чате, заданном в from_chat_id
- `caption` (`str`) – Новая подпись для медиа, 0-1024 символа после форматирования. Если не задано, используется исходная подпись
- `parse_mode` (`str`) – Режим форматирования новой подписи.
- `caption_entities` (Array of *`telebot.types.MessageEntity`*) – Список отформатированных частей новой подписи в формате JSON, можно использовать вместо parse_mode
- `disable_notification` (`bool`) – Отправить сообщение, при получении которого пользователи получат уведомление без звука.
- `protect_content` (`bool`) – Запретить пересылку и сохранение содержимого сообщения

- `reply_to_message_id` (`int`) – Deprecated - Use reply_parameters instead. If the message is a reply, ID of the original message
- `allow_sending_without_reply` (`bool`) – Deprecated - Use reply_parameters instead. 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*) – Дополнительные элементы интерфейса. Inline клавиатура, текстовая клавиатура, запрос на удаление текстовой клавиатуры или запрос на ответ от пользователя.
- `timeout` (`int`) – Таймаут запроса в секундах.
- `message_thread_id` (`int`) – id топика, в который нужно отправить сообщение
- `reply_parameters` (*telebot.types.ReplyParameters*) – Reply parameters.

Результат

On success, the MessageId of the sent message is returned.

Тип результата

telebot.types.MessageID

`async copy_messages(`*chat_id: str | int*, *from_chat_id: str | int*, *message_ids: List[int]*, *disable_notification: bool | None = None*, *message_thread_id: int | None = None*, *protect_content: bool | None = None*, *remove_caption: bool | None = None*`)` → List[*MessageID*]

Use this method to copy messages of any kind. If some of the specified messages can't be found or copied, they are skipped. Service messages, giveaway messages, giveaway winners messages, and invoice messages can't be copied. A quiz poll can be copied only if the value of the field correct_option_id is known to the bot. The method is analogous to the method forwardMessages, but the copied messages don't have a link to the original message. Album grouping is kept for copied messages. On success, an array of MessageId of the sent messages is returned.

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

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `from_chat_id` (`int` or `str`) – Уникальный id чата, в который было отправлено исходное сообщение (или username канала в формате @channelusername)
- `message_ids` (`list` of `int`) – Message identifiers in the chat specified in from_chat_id
- `disable_notification` (`bool`) – Отправить сообщение, при получении которого пользователи получат уведомление без звука
- `message_thread_id` (`int`) – id топика, в который будет отправлена группа медиа
- `protect_content` (`bool`) – Запретить пересылку и сохранение содержимого пересланного сообщения
- `remove_caption` (`bool`) – Pass True to copy the messages without their captions

Результат

On success, an array of MessageId of the sent messages is returned.

Тип результата
list of *telebot.types.MessageID*

async create_chat_invite_link(*chat_id: int | str*, *name: str | None = None*, *expire_date: int | datetime | None = None*, *member_limit: int | None = None*, *creates_join_request: bool | None = None*) → *ChatInviteLink*

Используйте этот метод, чтобы создать дополнительную ссылку-приглашение в чат. Бот должен быть администратором чата и иметь соответствующие права администратора. Ссылка может быть аннулирована методом revokeChatInviteLink. Возвращает новую ссылку-приглашение (ChatInviteLink).

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

Параметры

- chat_id (int or str) – Уникальный id чата или username канала (в формате @channelusername)
- name (str) – Название ссылки-приглашения; 0-32 символа
- expire_date (int or datetime) – Время, когда ссылка будет аннулирована в формате Unix timestamp
- member_limit (int) – Максимальное количество пользователей в чате
- creates_join_request (bool) – True, если пользователи, использующие эту ссылку должны быть одобрены администраторами чата. Нельзя использовать True вместе с member_limit

Результат
Возвращает новую ссылку-приглашение (ChatInviteLink).

Тип результата
telebot.types.ChatInviteLink

async create_forum_topic(*chat_id: int*, *name: str*, *icon_color: int | None = None*, *icon_custom_emoji_id: str | None = None*) → *ForumTopic*

Используйте этот метод, чтобы создать топик в супергруппе. Бот должен быть администратором чата и иметь права администратора can_manage_topics. Возвращает информацию о созданном топике (ForumTopic).

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

Параметры

- chat_id (int or str) – Уникальный id чата или username канала (в формате @channelusername)
- name (str) – Имя топика, 1-128 символов
- icon_color (int) – Цвет иконки топика в формате RGB. В текущий момент, доступны цвета 0x6FB9F0, 0xFFD67E, 0xCB86DB, 0x8EEE98, 0xFF93B2, or 0xFB6F5F
- icon_custom_emoji_id (str) – Кастомный эмодзи для использования в качестве иконки топика. Должно быть “tgs” эмодзи и быть ровно 1 символом

Результат
В случае успеха возвращается информация о созданном топике (ForumTopic).

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: int | None = None*, *suggested_tip_amounts: List[int] | None = None*, *provider_data: str | None = None*, *photo_url: str | None = None*, *photo_size: int | None = None*, *photo_width: int | None = None*, *photo_height: int | None = None*, *need_name: bool | None = None*, *need_phone_number: bool | None = None*, *need_email: bool | None = None*, *need_shipping_address: bool | None = None*, *send_phone_number_to_provider: bool | None = None*, *send_email_to_provider: bool | None = None*, *is_flexible: bool | None = None*`)` → str

используйте этот метод, чтобы создать ссылку-инвойс. Возвращает созданную ссылку в случае успеха (String).

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

Параметры

- `title` (`str`) – Название товара, 1-32 символа
- `description` (`str`) – Описание товара, 1-255 символов
- `payload` (`str`) – Дополнительные данные, 1-128 байт. Не будет показано пользователю, используйте во внутренних процессах.
- `provider_token` (`str`) – Токен платежной системы, полученный через @BotFather
- `currency` (`str`) – Трехбуквенный код валюты в формате ISO 4217, см. https://core.telegram.org/bots/payments#supported-currencies
- `prices` (`list` of `types.LabeledPrice`) – Детали цены, список компонент (например цена продукта, налог, скидка, стоимость доставки, налог на доставку, бонус и т.д.)
- `max_tip_amount` (`int`) – Максимальный размер чаевых в наименьших единицах выбранной валюты
- `suggested_tip_amounts` (`list` of `int`) – Массив предлагаемых вариантов чаевых в наименьших единицах выбранной валюты в формате JSON. Можно задать не более 4 вариантов. Варианты чаевых должны быть больше нуля, перечисленные в порядке строгого возрастания и не превышать max_tip_amount.
- `provider_data` (`str`) – Данные о инвойсе в формате JSON, которые будут переданы платежной системе. Подробное описание обязательных полей должно быть предоставлено провайдером платежной системы.
- `photo_url` (`str`) – URL изображения товара для инвойса. Может быть изображением товаров или изображением инвойса. Людям больше нравится видеть фото товара, за который они платят.
- `photo_size` (`int`) – Вес изображения в байтах
- `photo_width` (`int`) – Ширина изображения
- `photo_height` (`int`) – Высота изображения
- `need_name` (`bool`) – Передайте True, если для совершения заказа требуется полное имя пользователя
- `need_phone_number` (`bool`) – Передайте True, если для совершения заказа требуется номер телефона пользователя

- `need_email` (`bool`) – Передайте True, если для совершения заказа требуется email пользователя
- `need_shipping_address` (`bool`) – Передайте True, если для совершения заказа требуется адрес доставки
- `send_phone_number_to_provider` (`bool`) – Передайте True, если номер телефона пользователя нужно отправить платежной системе
- `send_email_to_provider` (`bool`) – Передайте True, если email пользователя нужно отправить платежной системе
- `is_flexible` (`bool`) – Передайте True, если окончательная цена зависит от способа доставки

Результат
Созданная ссылка-инвойс (String) в случае успеха.

Тип результата
`str`

`async create_new_sticker_set(`*user_id: int, name: str, title: str, emojis: List[str] | None = None, png_sticker: Any | str = None, tgs_sticker: Any | str = None, webm_sticker: Any | str = None, contains_masks: bool | None = None, sticker_type: str | None = None, mask_position: MaskPosition | None = None, needs_repainting: bool | None = None, stickers: List[InputSticker] = None, sticker_format: str | None = None*`)` → bool

Используйте этот метод, чтобы создать новый стикерпак, владельцем которого станет пользователь. Бот будет иметь возможность редактировать созданный стикерпак. Возвращает True в случае успеха.

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

---

Примечание: Fields *_sticker are deprecated, pass a list of stickers to stickers parameter instead.

Параметры

- `user_id` (`int`) – id пользователя, создавшего стикерпак
- `name` (`str`) – Короткое имя стикерпака для использования в ссылках вида t.me/addstickers/ (например animals). Может содержать только латинские буквы, цифры и нижние подчеркивания. Должно начинаться с буквы, не может содержать подряд идущие нижние подчеркивания и должно заканчиваться на «_by_<bot_username>». <bot_username> учитывает регистр. 1-64 символа.
- `title` (`str`) – Название стикерпака, 1-64 символа
- `emojis` (`str`) – Один или несколько эмодзи, относящихся к стикеру
- `png_sticker` (`str`) – Изображение стикера в формате PNG, весом не более 512 килобайт, размеры не должны превышать 512px, либо ширина, либо высота должны быть ровно 512px. Передайте file_id в формате str, чтобы отправить уже загруженный на сервера Telegram файл, передайте HTTP URL в формате str, чтобы Telegram скачал файл из интернета, или загрузите новый файл с помощью multipart/form-data.

- `tgs_sticker` (`str`) – Анимированный стикер в формате TGS, загруженный с помощью multipart/form-data.
- `webm_sticker` (`str`) – Анимированный стикер в формате WebM, загруженный с помощью multipart/form-data.
- `contains_masks` (`bool`) – Передайте True, если создаётся стикерпак масок. Устарело, начиная с Bot API 6.2, используйте sticker_type.
- `sticker_type` (`str`) – Type of stickers in the set, pass "regular", "mask", or "custom_emoji". By default, a regular sticker set is created.
- `mask_position` (*`telebot.types.MaskPosition`*) – Позиция для размещения маски на лицах в формате JSON
- `needs_repainting` (`bool`) – Pass True if stickers in the sticker set must be repainted to the color of text when used in messages, the accent color if used as emoji status, white on chat photos, or another appropriate color based on context; for custom emoji sticker sets only
- `stickers` (`list` of *`telebot.types.InputSticker`*) – List of stickers to be added to the set
- `sticker_format` (`str`) – deprecated

Результат
В случае успеха возвращается True.

Тип результата
`bool`

`async decline_chat_join_request(`*chat_id: str | int*, *user_id: int | str*`)` → bool

Используйте этот метод, чтобы отклонить запрос на вступление в чат. Бот должен быть администратором чата и иметь права администратора can_invite_users. Возвращает True в случае успеха.

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

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username супергруппы (в формате @supergroupusername)
- `user_id` (`int` or `str`) – Уникальный id сделавшего запрос пользователя

Результат
True в случае успеха.

Тип результата
`bool`

`async delete_chat_photo(`*chat_id: int | str*`)` → bool

Используйте этот метод, чтобы удалить фото чата. Нельзя изменить фото в приватных чатах. Бот должен быть администратором чата и иметь соответствующие права администратора. Возвращает True в случае успеха. Примечание: В обычных группах (не супергруппах), метод будет работать только в случаях, когда настройка 'All Members Are Admins' выключена.

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

Параметры
`chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)

Результат
True в случае успеха.

Тип результата
`bool`

`async delete_chat_sticker_set(chat_id: int | str)` → bool

Используйте этот метод, чтобы удалить стикерпак группы из супергруппы. Бот должен быть администратором чата и иметь соответствующие права администратора. Используйте поле can_set_sticker_set, возвращаемое методом getChat, чтобы проверить, что бот может использовать этот метод. Возвращает True в случае успеха.

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

Параметры
`chat_id` (`int` or `str`) – Уникальный id чата или username супергруппы (в формате @supergroupusername)

Результат
Возвращает True в случае успеха.

Тип результата
`bool`

`async delete_forum_topic(chat_id: str | int, message_thread_id: int)` → bool

Используйте этот метод, чтобы удалить топик в супергруппе. Бот должен быть администратором чата и иметь права администратора can_manage_topics, за исключением случае, когда бот является создателем топика. Возвращает True в случае успеха.

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

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `message_thread_id` (`int`) – id топика, который нужно удалить

Результат
В случае успеха возвращается True.

Тип результата
`bool`

`async delete_message(chat_id: int | str, message_id: int, timeout: int | None = None)` → bool

Используйте этот метод, чтобы удалить сообщение, в том числе сервисное, ограничения: - Сообщение может быть удалено только если оно было отправлено менее 48 часов назад. - Dice-сообщение в приватном чате может быть удалено только если оно было отправлено более 24 часов назад. - Боты могут удалять свои сообщения в приватных чатах, группах и супергруппах. - Боты могут удалять чужие сообщение в приватных чатах. - Боты с правами администратора can_post_messages могут удалять сообщения в каналах. - Если бот является администратором группы, он может удалить любое сообщение в ней. - Если бот имеет права администратора can_delete_messages в супергруппе или канале, он может удалить любое сообщение в них. Возвращает True в случае успеха.

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

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)

- `message_id` (`int`) – id сообщения, которое нужно удалить
- `timeout` (`int`) – Таймаут запроса в секундах.

Результат
Возвращает True в случае успеха.

Тип результата
`bool`

`async delete_messages(`*chat_id: int | str*, *message_ids: List[int]*`)`

Use this method to delete multiple messages simultaneously. If some of the specified messages can't be found, they are skipped. Returns True on success.

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

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `message_ids` (`list` of `int`) – Identifiers of the messages to be deleted

Результат
Возвращает True в случае успеха.

`async delete_my_commands(`*scope: BotCommandScope | None = None*, *language_code: int | None = None*`)` → bool

Используйте этот метод, чтобы удалить список команд бота для заданных поля видимости и языка. После удаления, команды более широкого поля видимости будут доступны пользователям, которых коснулись изменения. Возвращает True в случае успеха.

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

Параметры

- `scope` (*`telebot.types.BotCommandScope`*) – Область видимости команд. По умолчанию BotCommandScopeDefault.
- `language_code` (`str`) – Двухбуквенный языковой код в формате ISO 639-1. Если не задан, изменения коснутся команд для всех пользователей в заданном поле видимости, не имеющих команд на их языке

Результат
True в случае успеха.

Тип результата
`bool`

`async delete_state(`*user_id: int*, *chat_id: int | None = None*`)`

Удалить текущее состояние (стейт) пользователя.

Параметры

- `user_id` (`int`) – id пользователя
- `chat_id` (`int`) – id чата

Результат
None

`async delete_sticker_from_set(sticker: str) → bool`

Используйте этот метод, чтобы удалить стикер из стикерпака, созданного ботом. Возвращает True в случае успеха.

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

Параметры
`sticker` – id файла стикера

Результат
В случае успеха возвращается True.

Тип результата
`bool`

`async delete_sticker_set(name: str) → bool`

Use this method to delete a sticker set. Returns True on success.

Параметры
`name` (`str`) – Имя стикерпака

Результат
Возвращает True в случае успеха.

Тип результата
`bool`

`async delete_webhook(drop_pending_updates: bool | None = None, timeout: int | None = None) → bool`

Используйте этот метод, чтобы удалить вебхук, если вы решите перейти обратно на getUpdates. Возвращает True в случае успеха.

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

Параметры

- `drop_pending_updates` – Передайте True, чтобы удалить все предшествующие запуску бота апдейты, по умолчанию None
- `timeout` (`int`, optional) – Тайм-аут запроса, по умолчанию None

Результат
Возвращает True в случае успеха.

Тип результата
`bool`

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

Handles new incoming deleted messages state.

Параметры

- `func` (`function`) – Функция, используемая в качестве фильтра
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`async download_file(file_path: str | None) → bytes`

Скачивает файл.

Параметры
`file_path` (*`str`*) – Путь, куда файл нужно сохранить.

Результат
bytes

Тип результата
`bytes`

`async edit_chat_invite_link(`*chat_id: int | str*, *invite_link: str | None = None*, *name: str | None = None*, *expire_date: int | datetime | None = None*, *member_limit: int | None = None*, *creates_join_request: bool | None = None*`) →` *ChatInviteLink*

Используйте этот метод, чтобы изменить неосновную ссылку-приглашение, созданную ботом. Бот должен быть администратором чата и иметь соответствующие права администратора.

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

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `name` (`str`) – Название ссылки-приглашения; 0-32 символа
- `invite_link` (`str`) – Ссылка-приглашение для изменения
- `expire_date` (`int` or `datetime`) – Время, когда ссылка будет аннулирована в формате Unix timestamp
- `member_limit` (`int`) – Максимальное количество пользователей в чате
- `creates_join_request` (`bool`) – True, если пользователи, использующие эту ссылку должны быть одобрены администраторами чата. Нельзя использовать True вместе с member_limit

Результат
Возвращает новую ссылку-приглашение (ChatInviteLink).

Тип результата
`telebot.types.ChatInviteLink`

`async edit_forum_topic(`*chat_id: int | str*, *message_thread_id: int*, *name: str | None = None*, *icon_custom_emoji_id: str | None = None*`) →` bool

Используйте этот метод, чтобы изменить название и иконку топика в супергруппе. Бот должен быть администратором чата и иметь права администратора can_manage_topics, кроме случаев, когда бот является создателем топика. Возвращает True в случае успеха.

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

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `message_thread_id` (`int`) – id топика для изменения
- `name` (`str`) – Необязательный, новое имя топика, 1-128 символов. Если не задано или пустое, сохранится текущее имя топика
- `icon_custom_emoji_id` (`str`) – Необязательный, новый уникальный id кастомного эмодзи, используемого в качестве иконки топика. Используйте getForumTopicIconStickers, чтобы получить все доступные id кастомных эмодзи. Передайте пустую строку, чтобы убрать иконку. Если не задан, сохранится текущая иконка топика

Результат
В случае успеха возвращается True.

Тип результата
`bool`

`async edit_general_forum_topic(`*chat_id: int | str*, *name: str*`)` → bool

Используйте этот метод, чтобы удалить топик в супергруппе. Бот должен быть администратором чата и иметь права администратора can_manage_topics, за исключением случае, когда бот является создателем топика. Возвращает True в случае успеха.

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

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `name` (`str`) – Название товара, 1-32 символа

`async edit_message_caption(`*caption: str*, *chat_id: int | str | None = None*, *message_id: int | None = None*, *inline_message_id: str | None = None*, *parse_mode: str | None = None*, *caption_entities: List[*MessageEntity*] | None = None*, *reply_markup:* InlineKeyboardMarkup *| None = None*`)` → *Message* | bool

Используйте этот метод, чтобы изменить подписи к медиа в сообщениях

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

Параметры

- `caption` (`str`) – Новая подпись к медиа
- `chat_id` (`int` | `str`) – Обязательный, если не указан inline_message_id. Уникальный id чата или username канала
- `message_id` (`int`) – Обязательный, если не указан inline_message_id.
- `inline_message_id` (`str`) – Обязательный, если не указан inline_message_id. id inline сообщения.
- `parse_mode` (`str`) – Новая подпись к медиа в сообщении, 0-1024 символа после форматирования
- `caption_entities` (`list` of `types.MessageEntity`) – Массив объектов, описывающих то, как будет происходить парсинг подписи к медиа в формате JSON.
- `reply_markup` (`InlineKeyboardMarkup`) – JSON-сериализованный объект inline клавиатуры.

Результат
В случае успеха, если изменённое сообщение отправлено ботом, возвращается новый объект Message, иначе (inline сообщения) возвращается True.

Тип результата
`types.Message` | `bool`

`async edit_message_live_location(`*latitude: float*, *longitude: float*, *chat_id: int | str | None = None*, *message_id: int | None = None*, *inline_message_id: str | None = None*, *reply_markup:* InlineKeyboardMarkup *| None = None*, *timeout: int | None = None*, *horizontal_accuracy: float | None = None*, *heading: int | None = None*, *proximity_alert_radius: int | None = None*`)` → *Message*

Используйте этот метод, чтобы изменить live местоположение в сообщении. Местоположение может быть изменено пока не истечёт live_period или не отключено вызовом метода stopMessageLiveLocation. В случае успеха, если измененное сообщение не является inline сообщением, возвращается новый объект Message, иначе возвращается True.

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

Параметры

- `latitude` (`float`) – Широта нового местоположения
- `longitude` (`float`) – Долгота нового местоположения
- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `message_id` (`int`) – Обязательный, если не указан inline_message_id. id сообщения, которое нужно изменить
- `reply_markup` (*`telebot.types.InlineKeyboardMarkup`* or *`telebot.types.ReplyKeyboardMarkup`* or *`telebot.types.ReplyKeyboardRemove`* or *`telebot.types.ForceReply`*) – JSON-сериализованный объект новой inline клавиатуры.
- `timeout` (`int`) – Таймаут запроса в секундах.
- `inline_message_id` (`str`) – Обязательный, если не указаны chat_id и message_id. id inline сообщения
- `horizontal_accuracy` (`float`) – Радиус погрешности местоположения, измеряется в метрах; 0-1500
- `heading` (`int`) – Направление, в котором пользователь двигается, в градусах. Если указано, должно быть от 1 до 360.
- `proximity_alert_radius` (`int`) – Максимальное расстояние для показа уведомлений о приближении других участников чата, в метрах. Если указано, должно быть от 1 до 100000.

Результат

В случае успеха, если измененное сообщение не является inline сообщением, возвращается новый объект Message, иначе возвращается True.

Тип результата

`telebot.types.Message` or bool

`async edit_message_media(`*media: Any, chat_id: int | str | None = None, message_id: int | None = None, inline_message_id: str | None = None, reply_markup:* InlineKeyboardMarkup *| None = None*`) →` *Message* | bool

Используйте этот метод, чтобы изменить гифку, аудио, документ, фото или видео в сообщении. Если сообщение является частью альбома, оно может быть изменено только на фото или видео. Иначе, тип сообщения может быть изменен на любой. При изменении inline сообщения, нельзя загрузить новый файл. используйте ранее загруженные файлы через file_id или укажите URL.

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

Параметры

- `media` (`InputMedia`) – JSON-сериализованный объект нового медиа контента

- `chat_id` (`int` or `str`) – Обязательный, если не указан inline_message_id. Уникальный id чата или username канала (в формате @channelusername)
- `message_id` (`int`) – Обязательный, если не указан inline_message_id. id отправленного сообщения
- `inline_message_id` (`str`) – Обязательный, если не указаны chat_id и message_id. id inline сообщения
- `reply_markup` (*telebot.types.InlineKeyboardMarkup* or `ReplyKeyboardMarkup` or `ReplyKeyboardRemove` or `ForceReply`) – JSON-сериализованный объект inline клавиатуры.

Результат

В случае успеха, если изменённое сообщение отправлено ботом, возвращается новый объект Message, иначе (inline сообщения) возвращается True.

Тип результата

`types.Message` or `bool`

`async edit_message_reply_markup(`*chat_id: int | str | None = None, message_id: int | None = None, inline_message_id: str | None = None, reply_markup:* InlineKeyboardMarkup *| None = None*`) →` *Message* | bool

Используйте этот метод, чтобы изменить только reply markup сообщения.

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

Параметры

- `chat_id` (`int` or `str`) – Обязательный, если не указан inline_message_id. Уникальный id чата или username канала (в формате @channelusername)
- `message_id` (`int`) – Обязательный, если не указан inline_message_id. id отправленного сообщения
- `inline_message_id` (`str`) – Обязательный, если не указаны chat_id и message_id. id inline сообщения
- `reply_markup` (`InlineKeyboardMarkup` or `ReplyKeyboardMarkup` or `ReplyKeyboardRemove` or `ForceReply`) – JSON-сериализованный объект inline клавиатуры.

Результат

В случае успеха, если изменённое сообщение отправлено ботом, возвращается новый объект Message, иначе (inline сообщения) возвращается True.

Тип результата

`types.Message` or `bool`

`async edit_message_text(`*text: str, chat_id: int | str | None = None, message_id: int | None = None, inline_message_id: str | None = None, parse_mode: str | None = None, entities: List[*MessageEntity*] | None = None, disable_web_page_preview: bool | None = None, reply_markup:* InlineKeyboardMarkup *| None = None, link_preview_options:* LinkPreviewOptions *| None = None*`) →` *Message* | bool

Используйте этот метод, чтобы изменить текстовые и игровые сообщения.

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

Параметры

- `text` (`str`) – Новый текст сообщения, 1-4096 символов после форматирования

- `chat_id` (`int` or `str`) – Обязательный, если не указан inline_message_id. Уникальный id чата или username канала (в формате @channelusername)
- `message_id` (`int`) – Обязательный, если не указан inline_message_id. id отправленного сообщения
- `inline_message_id` (`str`) – Обязательный, если не указаны chat_id и message_id. id inline сообщения
- `parse_mode` (`str`) – Режим форматирования в тексте сообщения.
- `entities` (List of *`telebot.types.MessageEntity`*) – Список отформатированных частей в тексте сообщения, можно использовать вместо parse_mode
- `disable_web_page_preview` (`bool`) – Deprecated - Use link_preview_options instead.
- `reply_markup` (`InlineKeyboardMarkup`) – JSON-сериализованный объект inline клавиатуры.
- `link_preview_options` (`LinkPreviewOptions`) – A JSON-serialized object for options used to automatically generate Telegram link previews for messages.

Результат

В случае успеха, если изменённое сообщение отправлено ботом, возвращается новый объект Message, иначе (inline сообщения) возвращается True.

Тип результата

`types.Message` or `bool`

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

Handles new version of a message(business accounts) 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`) – Необязательный список строк - команд для обработки.
- `regexp` (`str`) – Необязательное регулярное выражение.
- `func` (`function`) – Функция, используемая в качестве фильтра
- `content_types` (`list` of `str`) – Обрабатываемые виды контента. Обязан быть списком. По умолчанию [„text“]
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат

None

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

Обрабатывает новую версию поста в канале, который доступен боту и был изменён. В качестве параметра, передаёт в декорируемую функцию объект *`telebot.types.Message`*.

Параметры

- `commands` (`list` of `str`) – Необязательный список строк - команд для обработки.
- `regexp` (`str`) – Необязательное регулярное выражение.
- `func` (`function`) – Функция, используемая в качестве фильтра

- content_types (list of str) – Обрабатываемые виды контента. Обязан быть списком. По умолчанию [„text“]
- kwargs – Необязательные именованные аргументы(кастомные фильтры)

Результат

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

Обрабатывает новую версию сообщения, которое доступно боту и было изменено.

В качестве параметра, передаёт в декорируемую функцию объект *telebot.types.Message*.

Параметры

- commands (list of str) – Необязательный список строк - команд для обработки.
- regexp (str) – Необязательное регулярное выражение.
- func (function) – Функция, используемая в качестве фильтра
- content_types (list of str) – Обрабатываемые виды контента. Обязан быть списком. По умолчанию [„text“]
- chat_types (list of str) – список видов чатов
- kwargs – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

enable_saving_states(*filename='./.state-save/states.pkl'*)

Разрешить сохранение стейтов (по умолчанию сохранение отключено)

Примечание: Рекомендуется передавать экземпляр класса StatePickleStorage в качестве state_storage при инициализации класса TeleBot вместо использования этой функции.

Параметры
filename (str, optional) – Имя файла для сохранения, по умолчанию «./.state-save/states.pkl»

async export_chat_invite_link(*chat_id: int | str*) → str

Используйте этот метод, чтобы создать или заменить главную ссылку-приглашение в супергруппу или канал, созданную ботом. Бот должен быть администратором чата и иметь соответствующие права администратора.

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

Параметры
chat_id (int or str) – Уникальный id чата или username канала (в формате @channelusername)

Результат
новая ссылка-приглашение (String) в случае успеха.

Тип результата
str

`async forward_message(`*chat_id: int | str*, *from_chat_id: int | str*, *message_id: int*, *disable_notification: bool | None = None*, *protect_content: bool | None = None*, *timeout: int | None = None*, *message_thread_id: int | None = None*`)` → *Message*

Используйте этот метод, чтобы переслать любое сообщение.

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

Параметры

- `disable_notification` (`bool`) – Отправить сообщение, при получении которого пользователи получат уведомление без звука
- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `from_chat_id` (`int` or `str`) – Уникальный id чата, в который было отправлено исходное сообщение (или username канала в формате @channelusername)
- `message_id` (`int`) – id сообщения в чате, заданном в from_chat_id
- `protect_content` (`bool`) – Запретить пересылку и сохранение содержимого пересланного сообщения
- `timeout` (`int`) – Таймаут запроса в секундах.
- `message_thread_id` (`int`) – Уникальный id топика, в который нужно переслать сообщение; только для супергрупп с топиками

Результат

В случае успеха возвращает отправленное сообщение (Message).

Тип результата

`telebot.types.Message`

`async forward_messages(`*chat_id: str | int*, *from_chat_id: str | int*, *message_ids: List[int]*, *disable_notification: bool | None = None*, *message_thread_id: int | None = None*, *protect_content: bool | None = None*`)` → List[*MessageID*]

Use this method to forward multiple messages of any kind. If some of the specified messages can't be found or forwarded, they are skipped. Service messages and messages with protected content can't be forwarded. Album grouping is kept for forwarded messages. On success, an array of MessageId of the sent messages is returned.

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

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `from_chat_id` (`int` or `str`) – Уникальный id чата, в который было отправлено исходное сообщение (или username канала в формате @channelusername)
- `message_ids` (`list`) – Message identifiers in the chat specified in from_chat_id
- `disable_notification` (`bool`) – Отправить сообщение, при получении которого пользователи получат уведомление без звука
- `message_thread_id` (`int`) – id топика, в который будет отправлена группа медиа
- `protect_content` (`bool`) – Запретить пересылку и сохранение содержимого пересланного сообщения

Результат
В случае успеха возвращает отправленное сообщение (Message).

Тип результата
telebot.types.MessageID

`async get_business_connection(business_connection_id: str) → BusinessConnection`

Use this method to get information about the connection of the bot with a business account. Returns a BusinessConnection object on success.

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

Параметры
`business_connection_id` (`str`) – Unique identifier of the business connection

Результат
Returns a BusinessConnection object on success.

Тип результата
telebot.types.BusinessConnection

`async get_chat(chat_id: int | str) → Chat`

Используйте этот метод, чтобы получить актуальную информацию о чате (текущее имя пользователя для персональных диалогов, текущий username пользователя, группы или канала и т.д.). В случае успеха возвращает объект Chat.

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

Параметры
`chat_id` (`int` or `str`) – Уникальный id чата или username супергруппы или канала (в формате @channelusername)

Результат
Информация о чате

Тип результата
telebot.types.Chat

`async get_chat_administrators(chat_id: int | str) → List[ChatMember]`

Используйте этот метод, чтобы получить список администраторов чата. В случае успеха, возвращает массив объектов ChatMember, содержащих информацию обо всех администраторах чата, кроме других ботов.

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

Параметры
`chat_id` – Уникальный id чата или username супергруппы или канала (в формате @channelusername)

Результат
Список объектов ChatMember.

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

`async get_chat_member(chat_id: int | str, user_id: int) → ChatMember`

Используйте этот метод, чтобы получить информацию об участнике чата. Возвращает объект ChatMember в случае успеха.

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

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username супергруппы (в формате @supergroupusername)
- `user_id` (`int`) – Уникальный id сделавшего запрос пользователя

Результат
Возвращает объект ChatMember в случае успеха.

Тип результата
telebot.types.ChatMember

`async get_chat_member_count(chat_id: int | str)` → int

Используйте этот метод, чтобы получить количество участников чата.

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

Параметры
`chat_id` (`int` or `str`) – Уникальный id чата или username супергруппы или канала (в формате @channelusername)

Результат
Количество участников чата.

Тип результата
`int`

`get_chat_members_count(**kwargs)`

`async get_chat_menu_button(chat_id: int | str = None)` → *MenuButton*

Используйте этот метод, чтобы получить текущее значение кнопки menu в приватном чате, или кнопку menu по умолчанию. Возвращает MenuButton в случае успеха.

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

Параметры
`chat_id` (`int` or `str`) – Уникальный id приватного чата. Если не указан, будет возвращена кнопка menu по умолчанию.

Результат
types.MenuButton

Тип результата
telebot.types.MenuButton

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

Используйте этот метод, чтобы получить информацию о кастомных эмодзи по их id. Возвращает массив объектов Sticker.

Параметры
`custom_emoji_ids` (`list` of `str`) – Список id кастомных эмодзи. Можно указать не более 200 id.

Результат
Возвращает массив объектов Sticker.

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

`async get_file(file_id: str | None)` → *File*

Используйте этот метод, чтобы получить базовую информацию о файле и подготовить его к скачиванию. На текущий момент, боты могут скачивать файлы весом до 20MB. В случае успеха, возвращается объект File. Гарантируется, что ссылка на скачивание будет актуальна

как минимум 1 час. Когда ссылка перестаёт быть актуальной, новая может быть снова запрошена с помощью get_file.

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

Параметры
`file_id` (`str`) – id файла

Результат
`telebot.types.File`

`async get_file_url(`*file_id: str | None*`)` → str

Получить актуальную ссылку для скачивания файла.

Параметры
`file_id` (`str`) – id файла для получения ссылки на скачивание.

Результат
Ссылка для скачивания файла.

Тип результата
`str`

`async get_forum_topic_icon_stickers()` → List[*Sticker*]

Используйте этот метод, чтобы получить кастомные эмодзи, которые могут быть использованы любыми пользователями в качестве иконок топиков. Не требует параметров. Возвращает массив объектов Sticker.

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

Результат
В случае успеха, возвращается список объектов StickerSet.

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

`async get_game_high_scores(`*user_id: int, chat_id: int | str | None = None, message_id: int | None = None, inline_message_id: str | None = None*`)` → List[*GameHighScore*]

Используйте этот метод, чтобы получить данные для таблицы рекордов. Вернёт очки указанного пользователя и несколько соседних результатов. В случае успеха, возвращает массив объектов GameHighScore.

На текущий момент этот метод вернёт очки указанного пользователя и по два соседних результата с каждой стороны. Также вернет результаты трёх лучших игроков, если результат пользователя и соседние не являются тремя лучшими. Пожалуйста учитывайте, что это поведение может быть изменено.

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

Параметры

- `user_id` (`int`) – id пользователя
- `chat_id` (`int` or `str`) – Обязательный, если не указан inline_message_id. Уникальный id чата или username канала (в формате @channelusername)
- `message_id` (`int`) – Обязательный, если не указан inline_message_id. id отправленного сообщения
- `inline_message_id` (`str`) – Обязательный, если не указаны chat_id и message_id. id inline сообщения

Результат
В случае успеха, возвращает массив объектов GameHighScore.

Тип результата
List[*types.GameHighScore*]

`async get_me() → User`

Возвращает базовую информацию о боте в виде объекта User.

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

`async get_my_commands(scope: BotCommandScope | None, language_code: str | None) → List[BotCommand]`

Используйте этот метод, чтобы получить текущий список команд бота. Возвращает список объектов BotCommand в случае успеха.

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

Параметры

- `scope` (*telebot.types.BotCommandScope*) – Область видимости команд. По умолчанию BotCommandScopeDefault.
- `language_code` (`str`) – Двухбуквенный языковой код в формате ISO 639-1. Если не задан, изменения коснутся команд для всех пользователей в заданном поле видимости, не имеющих команд на их языке

Результат
Список объектов BotCommand в случае успеха.

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

`async get_my_default_administrator_rights(for_channels: bool = None) → ChatAdministratorRights`

Используйте этот метод, чтобы получить текущие права администратора для бота по умолчанию. Возвращает объект ChatAdministratorRights в случае успеха.

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

Параметры
`for_channels` (`bool`) – Передайте True, чтобы получить права администратора для бота по умолчанию в каналах. Иначе, будут возвращены права администратора для бота по умолчанию в группах и супергруппах.

Результат
Возвращает объект ChatAdministratorRights в случае успеха.

Тип результата
telebot.types.ChatAdministratorRights

`async get_my_description(language_code: str | None = None)`

Use this method to get the current bot description for the given user language. Returns BotDescription on success.

Параметры
`language_code` (`str`) – A two-letter ISO 639-1 language code or an empty string

Результат
telebot.types.BotDescription

`async get_my_name(language_code: str | None = None)`

Use this method to get the current bot name for the given user language. Returns BotName on success.

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

Параметры
`language_code` (`str`) – Optional. A two-letter ISO 639-1 language code or an empty string

Результат
telebot.types.BotName

`async get_my_short_description(language_code: str | None = None)`

Use this method to get the current bot short description for the given user language. Returns BotShortDescription on success.

Параметры
`language_code` (`str`) – A two-letter ISO 639-1 language code or an empty string

Результат
telebot.types.BotShortDescription

`async get_state(user_id, chat_id: int | None = None)`

Получает текущее состояние (стейт) пользователя. Не рекомендуется использовать этот метод. Но это удобно для дебага.

Параметры

- `user_id` (`int`) – id пользователя
- `chat_id` (`int`) – id чата

Результат
состояние (стейт) пользователя

Тип результата
`int` or `str` or `telebot.types.State`

`async get_sticker_set(name: str) → StickerSet`

Используйте этот метод, чтобы получить стикерпак. В случае успеха, возвращается объект StickerSet.

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

Параметры
`name` (`str`) – Имя стикерпака

Результат
В случае успеха, возвращается объект StickerSet.

Тип результата
telebot.types.StickerSet

`async get_updates(offset: int | None = None, limit: int | None = None, timeout: int | None = 20, allowed_updates: List | None = None, request_timeout: int | None = None) → List[Update]`

Используйте этот метод, чтобы получить новые апдейты с помощью long polling-а (wiki). Возвращается массив объектов Update.

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

Параметры

- `offset` (`int`, optional) – id первого апдейта. Должен быть на единицу больше наибольшего id среди ранее полученных апдейтов. По умолчанию, возвращается список апдейтов, начиная с самого раннего неполученного. Апдейт считается полученным как только вызван метод getUpdates со смещением больше, чем id этого апдейта. Отрицательное смещение может быть указано для получения последних offset апдейтов. Все предыдущие апдейты будут считаться полученными.
- `limit` (`int`, optional) – Максимальное число апдейтов для получения. Допускаются значения от 1 до 100. По умолчанию 100.
- `timeout` (`int`, optional) – Тайм-аут запроса
- `allowed_updates` (`list`, optional) – Массив строк. Список видов апдейтов, которые вы хотите получать.
- `request_timeout` (`int`, optional) – Timeout in seconds for request.

Результат
Возвращается массив объектов Update.

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

`async get_user_chat_boosts(`*chat_id: int | str*, *user_id: int*`) →` *UserChatBoosts*

Use this method to get the list of boosts added to a chat by a user. Requires administrator rights in the chat. Returns a UserChatBoosts object.

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

Параметры

- `chat_id` (`int | str`) – Уникальный id чата или username канала
- `user_id` (`int`) – Уникальный id сделавшего запрос пользователя

Результат
On success, a UserChatBoosts object is returned.

Тип результата
`telebot.types.UserChatBoosts`

`async get_user_profile_photos(`*user_id: int*, *offset: int | None = None*, *limit: int | None = None*`) →` *UserProfilePhotos*

Используйте этот метод, чтобы получить список аватарок пользователя. Возвращает объект *`telebot.types.UserProfilePhotos`*.

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

Параметры

- `user_id` (`int`) – Уникальный id сделавшего запрос пользователя
- `offset` (`int`) – Порядковый номер первого фото для получения. По умолчанию, возвращаются все фото.
- `limit` (`int`) – Максимальное число фото для получения. Допускаются значения от 1 до 100. По умолчанию 100.

Результат
UserProfilePhotos

Тип результата
`telebot.types.UserProfilePhotos`

`async get_webhook_info(timeout: int | None = None) → WebhookInfo`

Используйте этот метод, чтобы получить текущий статус вебхука. Не требует параметров. В случае успеха возвращает объект WebhookInfo. Если бот использует getUpdates, вернёт объект с пустым атрибутом url.

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

Параметры
`timeout` (`int`, optional) – Тайм-аут запроса

Результат
В случае успеха, возвращает объект WebhookInfo.

Тип результата
`telebot.types.WebhookInfo`

`async hide_general_forum_topic(chat_id: int | str) → bool`

Используйте этот метод, чтобы удалить топик в супергруппе. Бот должен быть администратором чата и иметь права администратора can_manage_topics, за исключением случае, когда бот является создателем топика. Возвращает True в случае успеха.

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

Параметры
`chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)

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

Запустить поллинг в бесконечном цикле с обработкой исключений, чтобы избежать непредвиденных остановок поллинга.

Примечание: Установите watchdog и psutil, чтобы использовать restart_on_change.

Параметры

- `timeout` (`int`) – Тайм-аут get_updates в секундах(по умолчанию None)
- `skip_pending` (`bool`) – пропускать старые апдейты
- `request_timeout` (`int`) – Тайм-аут запроса aiohttp. По умолчанию 5 минут(aiohttp.ClientTimeout).
- `logger_level` (`int`) – Кастомный уровень логирования для infinity_polling. Используйте уровни из logging в качестве значений. None/NOTSET = не логировать ошибки.
- `allowed_updates` (`list` of `str`) – Список видов апдейтов, которые вы хотите получать. Например, укажите ["message", "edited_channel_post", "callback_query"], чтобы получать апдейты только этих видов. Полный список доступных видов апдейтов - util.update_types. Укажите пустой список, чтобы получать все апдейты, кроме chat_member (по умолчанию). Если не задан, будет использована последняя настройка.

  Пожалуйста учитывайте, что этот параметр не влияет на апдейты, отправленные до вызова get_updates, поэтому нежелательные апдейты могут быть получены в течение короткого периода времени.

- `restart_on_change` (`bool`) – Перезапуск при изменении файлов. По умолчанию False
- `path_to_watch` (`str`) – Путь для мониторинга изменений. По умолчанию текущая директория.

Результат
None

`inline_handler(func, **kwargs)`

Обрабатывает inline query. В качестве параметра, передаёт в декорируемую функцию объект *telebot.types.InlineQuery*.

Параметры

- `func` (`function`) – Функция, используемая в качестве фильтра
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`async kick_chat_member(chat_id: int | str, user_id: int, until_date: int | datetime | None = None, revoke_messages: bool | None = None) → bool`

Эта функция устарела. Используйте *ban_chat_member*

`async leave_chat(chat_id: int | str) → bool`

Используйте этот метод, чтобы покинуть группу, супергруппу или канал. Возвращает True в случае успеха.

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

Параметры
`chat_id` (`int` or `str`) – Уникальный id чата или username супергруппы или канала (в формате @channelusername)

Результат
`bool`

`async log_out() → bool`

Используйте этот метод, чтобы отключиться от облачного Bot API сервера перед локальным запуском бота. Вы ДОЛЖНЫ отключить бота перед тем, как запускать его локально, иначе нет никаких гарантий, что бот будет получать апдейты. После успешного вызова, вы можете тут же подключиться к локальному серверу, но не сможете подключиться обратно к облачному Bot API серверу в течение 10 минут. Возвращает True в случае успеха.

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

Результат
True в случае успеха.

Тип результата
`bool`

`message_handler(commands=None, regexp=None, func=None, content_types=None, chat_types=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.

Пример:

Список 26: Использование 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`) – Необязательный список строк - команд для обработки.
- `regexp` (`str`) – Необязательное регулярное выражение.
- `func` – Необязательная lambda функция. Получает сообщение (объект Message) в качестве первого параметра. Функция должна вернуть True если хендлер должен обработать сообщение.
- `content_types` (`list` of `str`) – Обрабатываемые виды контента. Обязан быть списком. По умолчанию [„text“]
- `chat_types` (`list` of `str`) – список видов чатов
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат

декорируемая функция

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

Handles new incoming message reaction count. As a parameter to the decorator function, it passes *`telebot.types.MessageReactionCountUpdated`* object.

Параметры

- `func` (`function`) – Функция, используемая в качестве фильтра
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат

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

Handles new incoming message reaction. As a parameter to the decorator function, it passes *telebot.types.MessageReactionUpdated* object.

Параметры

- `func` (`function`) – Функция, используемая в качестве фильтра
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат

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

Обрабатывает изменения статуса бота. Для приватных чатов, этот апдейт отправляется только когда бот был заблокирован или разблокирован пользователем. В качестве параметра передаёт в декорируемую функцию объект *telebot.types.ChatMemberUpdated*.

Параметры

- `func` (`function`) – Функция, используемая в качестве фильтра
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`async pin_chat_message(chat_id: int | str, message_id: int, disable_notification: bool | None = False)` → bool

Используйте этот метод, чтобы закрепить сообщение в супергруппе. Бот должен быть администратором чата и иметь соответствующие права администратора. Возвращает True в случае успеха.

Документация Telegram: https://core.telegram.org/bots/api#pinchatmessage

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `message_id` (`int`) – id сообщения, которое нужно закрепить
- `disable_notification` (`bool`) – Передайте True, если всем участникам группы необходимо отправить уведомление о закреплённом сообщении

Результат
True в случае успеха.

Тип результата
`bool`

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

Обрабатывает изменения ответа пользователя в не анонимном опросе(когда пользователь меняет выбор). Боты получают новые ответы только в опросах, которые отправили сами. В качестве параметра передаёт в декорируемую функцию объект *telebot.types.PollAnswer*.

Параметры

- `func` (`function`) – Функция, используемая в качестве фильтра
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`poll_handler(func, **kwargs)`

Обрабатывает изменения в состоянии опроса. Боты получают только апдейты о завершенных опросах и опросах, которые отправили сами. В качестве параметра передаёт в декорируемую функцию объект `telebot.types.Poll`.

Параметры

- `func` (`function`) – Функция, используемая в качестве фильтра
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`async polling(non_stop: bool = False, skip_pending=False, interval: int = 0, timeout: int = 20, request_timeout: int | None = None, allowed_updates: List[str] | None = None, none_stop: bool | None = None, restart_on_change: bool | None = False, path_to_watch: str | None = None)`

Запускает бота в режиме поллинга в основном цикле событий. Это позволяет боту получать апдейты (Update) автоматически и вызывать соответствующие листенеры и хендлеры.

Предупреждение: Не вызывайте эту функцию более одного раза!

Всегда получает апдейты.

Примечание: Укажите non_stop=True, если хотите чтобы ваш бот продолжать получать апдейты при возникновении ошибок.

Примечание: Установите watchdog и psutil, чтобы использовать restart_on_change.

Параметры

- `non_stop` (`bool`) – Не останавливать поллинг при возникновении ApiException.
- `skip_pending` (`bool`) – пропускать старые апдейты
- `interval` (`int`) – Задержка между получением апдейтов
- `timeout` (`int`) – Тайм-аут запроса
- `request_timeout` (`int`) – Тайм-аут get_updates в секундах(по умолчанию None)
- `allowed_updates` (`list of str`) – Список видов апдейтов, которые вы хотите получать. Например, укажите ["message", "edited_channel_post", "callback_query"], чтобы получать апдейты только этих видов. Полный список доступных видов апдейтов - util.update_types. Укажите пустой список, чтобы получать все апдейты, кроме chat_member (по умолчанию). Если не задан, будет использована последняя настройка.

  Пожалуйста учитывайте, что этот параметр не влияет на апдейты, отправленные до вызова get_updates, поэтому нежелательные апдейты могут быть получены в течение короткого периода времени.

- `none_stop` (`bool`) – Устарело, используйте non_stop. Старая опечатка, оставлено для обратной совместимости
- `restart_on_change` (`bool`) – Перезапускать при изменениях в файлах. По умолчанию False.
- `path_to_watch` (`str`) – Путь для мониторинга изменений. По умолчанию текущая директория.

Результат

`pre_checkout_query_handler(`*func*, ***kwargs*`)`

Новая pre-checkout query. Содержит полную информацию о заказе. В качестве параметра передаёт в декорируемую функцию объект *`telebot.types.PreCheckoutQuery`*.

Параметры

- `func` (`function`) – Функция, используемая в качестве фильтра
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`async process_new_updates(`*updates: List[Update]*`)`

Обрабатывает новые апдейты. Просто передайте список апдейтов(Update и его наследники).

Параметры
`updates` (`list` of *`telebot.types.Update`*) – список апдейтов

Результат
None

`async promote_chat_member(`*chat_id: int | str, user_id: int, can_change_info: bool | None = None, can_post_messages: bool | None = None, can_edit_messages: bool | None = None, can_delete_messages: bool | None = None, can_invite_users: bool | None = None, can_restrict_members: bool | None = None, can_pin_messages: bool | None = None, can_promote_members: bool | None = None, is_anonymous: bool | None = None, can_manage_chat: bool | None = None, can_manage_video_chats: bool | None = None, can_manage_voice_chats: bool | None = None, can_manage_topics: bool | None = None, can_post_stories: bool | None = None, can_edit_stories: bool | None = None, can_delete_stories: bool | None = None*`)` → bool

Используйте этот метод, чтобы повысить или понизить пользователя в супергруппе или канале. Бот должен быть администратором чата и иметь соответствующие права администратора. Передайте False во все boolean параметры, чтобы понизить пользователя.

Документация Telegram: https://core.telegram.org/bots/api#promotechatmember

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `user_id` (`int`) – Уникальный id сделавшего запрос пользователя
- `can_change_info` (`bool`) – Передайте True, если администратор может менять название чата, аватарку и другие настройки

- `can_post_messages` (`bool`) – Передайте True, если администратор может создавать посты в канале, только для каналов
- `can_edit_messages` (`bool`) – Передайте True, если администратор может изменять сообщения других пользователей, только для каналов
- `can_delete_messages` (`bool`) – Передайте True, если администратор может удалять сообщения других пользователей
- `can_invite_users` (`bool`) – Передайте True, если администратор может приглашать новых пользователей в чат
- `can_restrict_members` (`bool`) – Передайте True, если администратор может ограничивать, банить или разбанивать участников чата
- `can_pin_messages` (`bool`) – Передайте True, если администратор может закреплять сообщения, только для супергрупп
- `can_promote_members` (`bool`) – Передайте True, если администратор может добавлять новых администраторов с подмножеством его собственных прав администратора или понижать администраторов, которых он повысил, напрямую или косвенно (администраторами, которых он назначил)
- `is_anonymous` (`bool`) – Передайте True, если присутствие администратора в чате скрыто
- `can_manage_chat` (`bool`) – Передайте True, если администратор имеет доступ к логу событий чата, статистике чата, статистике сообщений в каналах, видеть участников канала, видеть анонимных администраторов в супергруппах и игнорировать медленный режим. Подразумевается любым другим правом администратора
- `can_manage_video_chats` (`bool`) – Передайте True, если администратор может управлять голосовыми чатами. На текущий момент, боты могут использовать это право администратора только для передачи другим администраторам.
- `can_manage_voice_chats` (`bool`) – Устарело, используйте can_manage_video_chats.
- `can_manage_topics` (`bool`) – Передайте True, если пользователю разрешено создавать, переименовывать, закрывать, и возобновлять топики, только для супергрупп
- `can_post_stories` (`bool`) – Pass True if the administrator can create the channel's stories
- `can_edit_stories` (`bool`) – Pass True if the administrator can edit the channel's stories
- `can_delete_stories` (`bool`) – Pass True if the administrator can delete the channel's stories

Результат

True в случае успеха.

Тип результата

`bool`

`register_business_connection_handler(`*callback: Callable*, *func: Callable | None = None*, *pass_bot: bool | None = False*, ***kwargs*`)`

Registers business connection handler.

Параметры

- `callback` (`function`) – функция-хендлер
- `func` (`function`) – Функция, используемая в качестве фильтра
- `pass_bot` (`bool`) – True, если вам нужно передать экземпляр класса TeleBot в хендлер(удобно для разбиения кода на файлы)
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`register_business_message_handler(`*callback: Callable, commands: List[str] | None = None, regexp: str | None = None, func: Callable | None = None, content_types: List[str] | None = None, **kwargs*`)`

Registers business connection handler.

Параметры

- `callback` (`function`) – функция-хендлер
- `func` (`function`) – Функция, используемая в качестве фильтра
- `pass_bot` (`bool`) – True, если вам нужно передать экземпляр класса TeleBot в хендлер(удобно для разбиения кода на файлы)
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`register_callback_query_handler(`*callback: Callable[[Any], Awaitable], func: Callable, pass_bot: bool | None = False, **kwargs*`)`

Регистрирует хендлер callback query.

Параметры

- `callback` (`Awaitable`) – функция-хендлер
- `func` (`function`) – Функция, используемая в качестве фильтра
- `pass_bot` (`bool`) – True, если вам нужно передать экземпляр класса TeleBot в хендлер(удобно для разбиения кода на файлы)
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`register_channel_post_handler(`*callback: Callable[[Any], Awaitable], content_types: List[str] | None = None, commands: List[str] | None = None, regexp: str | None = None, func: Callable | None = None, pass_bot: bool | None = False, **kwargs*`)`

Регистрирует хендлер постов в каналах.

Параметры

- `callback` (`Awaitable`) – функция-хендлер
- `content_types` (`list` of `str`) – Обрабатываемые виды контента. Обязан быть списком. По умолчанию [„text“]
- `commands` (`list` of `str`) – список команд
- `regexp` (`str`) – Регулярное выражение

- `func` (`function`) – Функция, используемая в качестве фильтра
- `pass_bot` (`bool`) – True, если вам нужно передать экземпляр класса TeleBot в хендлер(удобно для разбиения кода на файлы)
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`register_chat_boost_handler(`*callback: Callable*, *func: Callable | None = None*, *pass_bot: bool | None = False*, ***kwargs*`)`

Registers chat boost handler.

Параметры

- `callback` (`function`) – функция-хендлер
- `func` (`function`) – Функция, используемая в качестве фильтра
- `pass_bot` – True, если вам нужно передать экземпляр класса TeleBot в хендлер(удобно для разбиения кода на файлы)
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`register_chat_join_request_handler(`*callback: Callable[[Any], Awaitable]*, *func: Callable | None = None*, *pass_bot: bool | None = False*, ***kwargs*`)`

Регистрирует хендлер запросов на вступление в чат.

Параметры

- `callback` (`Awaitable`) – функция-хендлер
- `func` (`function`) – Функция, используемая в качестве фильтра
- `pass_bot` (`bool`) – True, если вам нужно передать экземпляр класса TeleBot в хендлер(удобно для разбиения кода на файлы)
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`register_chat_member_handler(`*callback: Callable[[Any], Awaitable]*, *func: Callable | None = None*, *pass_bot: bool | None = False*, ***kwargs*`)`

Регистрирует хендлер смены состояний участников чата.

Параметры

- `callback` (`Awaitable``) – функция-хендлер
- `func` (`function`) – Функция, используемая в качестве фильтра
- `pass_bot` (`bool`) – True, если вам нужно передать экземпляр класса TeleBot в хендлер(удобно для разбиения кода на файлы)
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

:return:None

`register_chosen_inline_handler(`*callback: Callable[[Any], Awaitable]*, *func: Callable*, *pass_bot: bool | None = False*, ***kwargs*`)`

Регистрирует хендлер выбора результата inline query.

Параметры

- `callback` (`Awaitable`) – функция-хендлер
- `func` (`function`) – Функция, используемая в качестве фильтра
- `pass_bot` (`bool`) – True, если вам нужно передать экземпляр класса TeleBot в хендлер(удобно для разбиения кода на файлы)
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`register_deleted_business_messages_handler(`*callback: Callable*, *func: Callable | None = None*, *pass_bot: bool | None = False*, ***kwargs*`)`

Registers deleted business messages handler.

Параметры

- `callback` (`function`) – функция-хендлер
- `func` (`function`) – Функция, используемая в качестве фильтра
- `pass_bot` (`bool`) – True, если вам нужно передать экземпляр класса TeleBot в хендлер(удобно для разбиения кода на файлы)
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`register_edited_business_message_handler(`*callback: Callable*, *content_types: List[str] | None = None*, *commands: List[str] | None = None*, *regexp: str | None = None*, *func: Callable | None = None*, *pass_bot: bool | None = False*, ***kwargs*`)`

Registers edited message handler for business accounts.

Параметры

- `callback` (`function`) – функция-хендлер
- `content_types` (`list` of `str`) – Обрабатываемые виды контента. Обязан быть списком. По умолчанию [„text“]
- `commands` (`list` of `str`) – список команд
- `regexp` (`str`) – Регулярное выражение
- `func` (`function`) – Функция, используемая в качестве фильтра
- `pass_bot` (`bool`) – True, если вам нужно передать экземпляр класса TeleBot в хендлер(удобно для разбиения кода на файлы)
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`register_edited_channel_post_handler(`*callback: Callable[[Any], Awaitable]*, *content_types: List[str] | None = None*, *commands: List[str] | None = None*, *regexp: str | None = None*, *func: Callable | None = None*, *pass_bot: bool | None = False*, ***kwargs*`)`

Регистрирует хендлер изменения постов в каналах.

Параметры

- `callback` (`Awaitable`) – функция-хендлер
- `content_types` (`list` of `str`) – Обрабатываемые виды контента. Обязан быть списком. По умолчанию [„text“]
- `commands` (`list` of `str`) – список команд
- `regexp` (`str`) – Регулярное выражение
- `func` (`function`) – Функция, используемая в качестве фильтра
- `pass_bot` (`bool`) – True, если вам нужно передать экземпляр класса TeleBot в хендлер(удобно для разбиения кода на файлы)
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат

декорируемая функция

`register_edited_message_handler(`*callback: Callable[[Any], Awaitable]*, *content_types: List[str] | None = None*, *commands: List[str] | None = None*, *regexp: str | None = None*, *func: Callable | None = None*, *chat_types: List[str] | None = None*, *pass_bot: bool | None = False*, ***kwargs*`)`

Регистрирует хендлер изменения сообщений.

Параметры

- `callback` (`Awaitable`) – функция-хендлер
- `content_types` (`list` of `str`) – Обрабатываемые виды контента. Обязан быть списком. По умолчанию [„text“]
- `commands` (`list` of `str`) – список команд
- `regexp` (`str`) – Регулярное выражение
- `func` (`function`) – Функция, используемая в качестве фильтра
- `chat_types` (`bool`) – True для приватных чатов
- `pass_bot` (`bool`) – True, если вам нужно передать экземпляр класса TeleBot в хендлер(удобно для разбиения кода на файлы)
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат

None

`register_inline_handler(`*callback: Callable[[Any], Awaitable]*, *func: Callable*, *pass_bot: bool | None = False*, ***kwargs*`)`

Регистрирует хендлер inline query.

Параметры

- `callback` (`Awaitable`) – функция-хендлер

- `func` (`function`) – Функция, используемая в качестве фильтра
- `pass_bot` (`bool`) – True, если вам нужно передать экземпляр класса TeleBot в хендлер(удобно для разбиения кода на файлы)
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
декорируемая функция

`register_message_handler(`*callback: Callable[[Any], Awaitable]*, *content_types: List[str] | None = None*, *commands: List[str] | None = None*, *regexp: str | None = None*, *func: Callable | None = None*, *chat_types: List[str] | None = None*, *pass_bot: bool | None = False*, ***kwargs*`)`

Регистрирует хендлер сообщений.

Параметры

- `callback` (`Awaitable`) – функция-хендлер
- `content_types` (`list` of `str`) – Обрабатываемые виды контента. Обязан быть списком. По умолчанию [„text“]
- `commands` (`list` of `str`) – список команд
- `regexp` (`str`)
- `func` (`function`) – Функция, используемая в качестве фильтра
- `chat_types` (`list` of `str`) – Список видов чатов
- `pass_bot` (`bool`) – True, если вам нужно передать экземпляр класса TeleBot в хендлер(удобно для разбиения кода на файлы)
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`register_message_reaction_count_handler(`*callback: Callable[[Any], Awaitable]*, *func: Callable = None*, *pass_bot: bool | None = False*, ***kwargs*`)`

Registers message reaction count handler.

Параметры

- `callback` (`Awaitable`) – функция-хендлер
- `func` (`function`) – Функция, используемая в качестве фильтра
- `pass_bot` (`bool`) – True, если вам нужно передать экземпляр класса TeleBot в хендлер(удобно для разбиения кода на файлы)
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`register_message_reaction_handler(`*callback: Callable[[Any], Awaitable]*, *func: Callable = None*, *pass_bot: bool | None = False*, ***kwargs*`)`

Registers message reaction handler.

Параметры

- `callback` (`Awaitable`) – функция-хендлер

- `func` (`function`) – Функция, используемая в качестве фильтра
- `pass_bot` (`bool`) – True, если вам нужно передать экземпляр класса TeleBot в хендлер(удобно для разбиения кода на файлы)
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`register_my_chat_member_handler(`*callback: Callable[[Any], Awaitable]*, *func: Callable | None = None*, *pass_bot: bool | None = False*, ***kwargs*`)`

Регистрирует хендлер изменений статуса бота.

Параметры

- `callback` (`Awaitable`) – функция-хендлер
- `func` (`function`) – Функция, используемая в качестве фильтра
- `pass_bot` (`bool`) – True, если вам нужно передать экземпляр класса TeleBot в хендлер(удобно для разбиения кода на файлы)
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`register_poll_answer_handler(`*callback: Callable[[Any], Awaitable]*, *func: Callable*, *pass_bot: bool | None = False*, ***kwargs*`)`

Регистрирует хендлер ответов в опросах.

Параметры

- `callback` (`Awaitable`) – функция-хендлер
- `func` (`function`) – Функция, используемая в качестве фильтра
- `pass_bot` (`bool`) – True, если вам нужно передать экземпляр класса TeleBot в хендлер(удобно для разбиения кода на файлы)
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`register_poll_handler(`*callback: Callable[[Any], Awaitable]*, *func: Callable*, *pass_bot: bool | None = False*, ***kwargs*`)`

Регистрирует хендлер изменений состояния опросов.

Параметры

- `callback` (`Awaitable`) – функция-хендлер
- `func` (`function`) – Функция, используемая в качестве фильтра
- `pass_bot` (`bool`) – True, если вам нужно передать экземпляр класса TeleBot в хендлер(удобно для разбиения кода на файлы)
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`register_pre_checkout_query_handler(`*callback: Callable[[Any], Awaitable]*, *func: Callable*, *pass_bot: bool | None = False*, ***kwargs*`)`

Регистрирует хендлер pre-checkout query.

Параметры

- `callback` (`Awaitable`) – функция-хендлер
- `func` – Функция, используемая в качестве фильтра
- `pass_bot` (`bool`) – True, если вам нужно передать экземпляр класса TeleBot в хендлер(удобно для разбиения кода на файлы)
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат

декорируемая функция

`register_removed_chat_boost_handler(`*callback: Callable*, *func: Callable | None = None*, *pass_bot: bool | None = False*, ***kwargs*`)`

Registers removed chat boost handler.

Параметры

- `callback` (`function`) – функция-хендлер
- `func` (`function`) – Функция, используемая в качестве фильтра
- `pass_bot` – True, если вам нужно передать экземпляр класса TeleBot в хендлер(удобно для разбиения кода на файлы)
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат

None

`register_shipping_query_handler(`*callback: Callable[[Any], Awaitable]*, *func: Callable*, *pass_bot: bool | None = False*, ***kwargs*`)`

Регистрирует хендлер shipping query.

Параметры

- `callback` (`Awaitable`) – функция-хендлер
- `func` (`function`) – Функция, используемая в качестве фильтра
- `pass_bot` (`bool`) – True, если вам нужно передать экземпляр класса TeleBot в хендлер(удобно для разбиения кода на файлы)
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат

None

`async remove_webhook()` → bool

Альтернатива delete_webhook, но использует set_webhook

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

Handles new incoming chat boost state. it passes *`telebot.types.ChatBoostRemoved`* object.

Параметры

- `func` (`function`) – Функция, используемая в качестве фильтра
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`async reopen_forum_topic(chat_id: str | int, message_thread_id: int)` → bool

Используйте этот метод, чтобы возобновить закрытый топик в супергруппе с топиками. Бот должен быть администратором чата и иметь права администратора can_manage_topics, кроме случаев, когда бот является создателем топика. Возвращает True в случае успеха.

Документация Telegram: https://core.telegram.org/bots/api#reopenforumtopic

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `message_thread_id` (`int`) – id топика для возобновления

Результат
В случае успеха возвращается True.

Тип результата
`bool`

`async reopen_general_forum_topic(chat_id: int | str)` → bool

Используйте этот метод, чтобы возобновить топик „General“ в супергруппе с топиками. Бот должен быть администратором чата и иметь права администратора can_manage_topics, кроме случаев, когда бот является создателем топика. Возвращает True в случае успеха.

Документация Telegram: https://core.telegram.org/bots/api#reopengeneralforumtopic

Параметры
`chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)

`async replace_sticker_in_set(user_id: int, name: str, old_sticker: str, sticker: InputSticker)` → bool

Use this method to replace an existing sticker in a sticker set with a new one. The method is equivalent to calling deleteStickerFromSet, then addStickerToSet,
then setStickerPositionInSet. Returns True on success.

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

Параметры

- `user_id` (`int`) – User identifier of the sticker set owner
- `name` (`str`) – Имя стикерпака
- `old_sticker` (`str`) – File identifier of the replaced sticker
- `sticker` (*`telebot.types.InputSticker`*) – A JSON-serialized object with information about the added sticker. If exactly the same sticker had already been added to the set, then the set remains unchanged.

Результат
Возвращает True в случае успеха.

Тип результата
`bool`

`async reply_to(message: Message, text: str, **kwargs)` → *Message*

Convenience function for *send_message(message.chat.id, text, reply_parameters=(message.message_id...), **kwargs)*

Параметры

- `message` (`types.Message`) – Экземпляр класса *telebot.types.Message*
- `text` (`str`) – Текст сообщения.
- `kwargs` – Дополнительные именованные аргументы, передаваемые в *telebot.TeleBot.send_message()*

Результат
В случае успеха возвращает отправленное сообщение (Message).

Тип результата
telebot.types.Message

`async reset_data(`*user_id: int*, *chat_id: int | None = None*`)`

Сбросить данные о пользователе в чате.

Параметры

- `user_id` (`int`) – id пользователя
- `chat_id` (`int`) – id чата

Результат
None

`async restrict_chat_member(`*chat_id: int | str*, *user_id: int*, *until_date: int | datetime | None = None*, *can_send_messages: bool | None = None*, *can_send_media_messages: bool | None = None*, *can_send_polls: bool | None = None*, *can_send_other_messages: bool | None = None*, *can_add_web_page_previews: bool | None = None*, *can_change_info: bool | None = None*, *can_invite_users: bool | None = None*, *can_pin_messages: bool | None = None*, *permissions:* ChatPermissions *| None = None*, *use_independent_chat_permissions: bool | None = None*`)` → bool

Используйте этот метод, чтобы ограничить пользователя в супергруппе. Бот должен быть администратором супергруппы и иметь соответствующие права администратора. Передайте True во все boolean параметры, чтобы снять с пользователя ограничения.

Документация Telegram: https://core.telegram.org/bots/api#restrictchatmember

> **Предупреждение:** Individual parameters are deprecated and will be removed, use „permissions“ instead

Параметры

- `chat_id` (`int` or `str`) – Уникальный id группы или username супергруппы или канала (в формате @channelusername)
- `user_id` (`int`) – Уникальный id сделавшего запрос пользователя
- `until_date` (`int` or `datetime`, optional) – Дата, когда ограничения будут сняты с пользователя, UNIX timestamp. Если пользователь ограничен более чем на 366 дней или менее чем на 30 секунд с текущего момента, он будет ограничен навсегда (пока ограничения не будут сняты вручную)
- `can_send_messages` (`bool`) – deprecated
- `can_send_media_messages` (`bool`) – deprecated

- `can_send_polls` (`bool`) – deprecated
- `can_send_other_messages` (`bool`) – deprecated
- `can_add_web_page_previews` (`bool`) – deprecated
- `can_change_info` (`bool`) – deprecated
- `can_invite_users` (`bool`) – deprecated
- `can_pin_messages` (`bool`) – deprecated
- `use_independent_chat_permissions` (`bool`, optional) – Pass True if chat permissions are set independently. Otherwise, the can_send_other_messages and can_add_web_page_previews permissions will imply the can_send_messages, can_send_audios, can_send_documents, can_send_photos, can_send_videos, can_send_video_notes, and can_send_voice_notes permissions; the can_send_polls permission will imply the can_send_messages permission.
- `permissions` (`types.ChatPermissions`) – Pass ChatPermissions object to set all permissions at once. Use this parameter instead of passing all boolean parameters to avoid backward compatibility problems in future.

Результат
True в случае успеха

Тип результата
`bool`

`retrieve_data`(*user_id: int*, *chat_id: int | None = None*)

Возвращает контекстный менеджер с данными о пользователе в чате.

Параметры

- `user_id` (*int*) – id пользователя
- `chat_id` (*int, optional*) – Уникальный id чата, по умолчанию user_id

Результат
Контекстный менеджер с данными о пользователе в чате.

Тип результата
Optional[Any]

`async revoke_chat_invite_link`(*chat_id: int | str*, *invite_link: str*) → *ChatInviteLink*

Используйте этот метод, чтобы аннулировать ссылку-приглашение, созданную ботом. Примечание: Если аннулируется главная ссылка-приглашение, автоматически генерируется новая. Бот должен быть администратором чата и иметь соответствующие права администратора.

Документация Telegram: https://core.telegram.org/bots/api#revokechatinvitelink

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `invite_link` (`str`) – Ссылка-приглашение, которую нужно аннулировать

Результат
Возвращает новую ссылку-приглашение (ChatInviteLink).

Тип результата
telebot.types.ChatInviteLink

`async run_webhooks(`*listen: str | None = '127.0.0.1', port: int | None = 443, url_path: str | None = None, certificate: str | None = None, certificate_key: str | None = None, webhook_url: str | None = None, max_connections: int | None = None, allowed_updates: List | None = None, ip_address: str | None = None, drop_pending_updates: bool | None = None, timeout: int | None = None, secret_token: str | None = None, secret_token_length: int | None = 20, debug: bool | None = False*`)`

Этот класс устанавливает вебхуки и мониторит указанный URL и порт.

Параметры

- `listen` – IP адрес для мониторинга. По умолчанию 0.0.0.0
- `port` – Порт, который будет использован для мониторинга вебхуков.
- `url_path` – Путь к вебхуку. По умолчанию /token.
- `certificate` – Путь к файлу с SSL сертификатом.
- `certificate_key` – Путь к файлу с приватным ключом SSL сертификата.
- `webhook_url` – URL вебхука.
- `max_connections` – Максимально-допустимое количество одновременных HTTPS подключений к вебхуку для доставки апдейтов, 1-100. По умолчанию 40. Используйте меньшие значения, чтобы уменьшить нагрузку на ваш сервер и большие значения для увеличения пропускной способности вашего бота.
- `allowed_updates` – Список видов апдейтов, которые вы хотите получать, в формате JSON. Например, укажите ["message", "edited_channel_post", "callback_query"], чтобы получать апдейты только этих видов. Полный список доступных видов апдейтов - util.update_types. Укажите пустой список, чтобы получать все апдейты, кроме chat_member (по умолчанию). Если не задан, будет использована последняя настройка.
- `ip_address` – Фиксированный IP адрес, который будет использоваться для отправки запросов к вебхуку вместо IP адреса, полученного через DNS
- `drop_pending_updates` – Передайте True, чтобы проигнорировать все апдейты, полученные до запуска
- `timeout` – Integer. Тайм-аут запроса на подключение.
- `secret_token` – Секретный токен для верификации запроса к вебхуку.
- `secret_token_length` – Length of a secret token, defaults to 20
- `debug` – Debug mode, defaults to False

Результат

`async send_animation(`*chat_id: int | str, animation: Any | str, duration: int | None = None, width: int | None = None, height: int | None = None, thumbnail: str | Any | None = None, caption: str | None = None, parse_mode: str | None = None, caption_entities: List[MessageEntity] | None = None, disable_notification: bool | None = None, protect_content: bool | None = None, reply_to_message_id: int | None = None, allow_sending_without_reply: bool | None = None, reply_markup: InlineKeyboardMarkup | ReplyKeyboardMarkup | ReplyKeyboardRemove | ForceReply | None = None, timeout: int | None = None, message_thread_id: int | None = None, has_spoiler: bool | None = None, thumb: str | Any | None = None, reply_parameters: ReplyParameters | None = None, business_connection_id: str | None = None*`) → *Message*

Используйте этот метод, чтобы отправить гифку (GIF или H.264/MPEG-4 AVC видео без звука). В случае успеха, возвращается отправленное сообщение (Message). На текущий момент, боты могут отправлять гифки весом до 50 MB, это ограничение может измениться в будущем.

Документация Telegram: https://core.telegram.org/bots/api#sendanimation

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `animation` (`str` or *`telebot.types.InputFile`*) – Гиф-ка для отправки. Передайте file_id (String), чтобы отправить гифку, которая уже загружена на сервера Telegram (рекомендуется), передайте HTTP URL (String), чтобы отправить гифку из интернета или загрузите новую гифку с помощью multipart/form-data.
- `duration` (`int`) – Длительность отправленной гифки в секундах
- `width` (`int`) – Ширина гифки
- `height` (`int`) – Высота гифки
- `thumbnail` (`str` or *`telebot.types.InputFile`*) – Обложка отправленного файла; может быть проигнорирована, если генерация обложки поддерживается на стороне сервера. Обложка должна быть картинкой в формате JPEG и весить менее 200 kB. Ширина и высота обложки не должны превышать 320. Игнорируется, если файл не загружен с помощью multipart/form-data. Обложки не могут быть использованы повторно и могут быть загружены только как новый файл, так что вы можете передать "attach://<file_attach_name>" если обложка была загружена с помощью multipart/form-data под именем <file_attach_name>.
- `caption` (`str`) – Подпись к гифке (может быть использована при повторной отправке гифки по file_id), 0-1024 символа после форматирования
- `parse_mode` (`str`) – Режим форматирования подписи к гифке
- `protect_content` (`bool`) – Запретить пересылку и сохранение содержимого сообщения
- `reply_to_message_id` (`int`) – Deprecated - Use reply_parameters instead. 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`*) – Дополнительные элементы интерфейса. Inline клавиатура, текстовая клавиатура, запрос на удаление текстовой клавиатуры или запрос на ответ от пользователя.
- `disable_notification` (`bool`) – Отправить сообщение, при получении которого пользователи получат уведомление без звука.
- `timeout` (`int`) – Таймаут запроса в секундах.
- `caption_entities` (`list` of *`telebot.types.MessageEntity`*) – Список отформатированных частей подписи, можно использовать вместо parse_mode

- `allow_sending_without_reply` (`bool`) – Deprecated - Use reply_parameters instead. Pass True, if the message should be sent even if the specified replied-to message is not found
- `message_thread_id` (`int`) – id топика, в который будет отправлено видео
- `has_spoiler` (`bool`) – Передайте True, если гифку нужно отправить как спойлер
- `thumb` (`str` or *telebot.types.InputFile*) – Deprecated. Use thumbnail instead
- `reply_parameters` (*telebot.types.ReplyParameters*) – Reply parameters.
- `business_connection_id` (`str`) – Identifier of a business connection, in which the message will be sent

Результат

В случае успеха возвращает отправленное сообщение (Message).

Тип результата

telebot.types.Message

`async send_audio(`*chat_id: int | str, audio: Any | str, caption: str | None = None, duration: int | None = None, performer: str | None = None, title: str | None = None, reply_to_message_id: int | None = None, reply_markup:* InlineKeyboardMarkup | ReplyKeyboardMarkup | ReplyKeyboardRemove | ForceReply *| None = None, parse_mode: str | None = None, disable_notification: bool | None = None, timeout: int | None = None, thumbnail: str | Any | None = None, caption_entities: List[*MessageEntity*] | None = None, allow_sending_without_reply: bool | None = None, protect_content: bool | None = None, message_thread_id: int | None = None, thumb: str | Any | None = None, reply_parameters:* ReplyParameters *| None = None, business_connection_id: str | None = None*`) → *Message*`

Используйте этот метод, чтобы отправить аудио, если вы хотите, чтобы клиенты (приложения) Telegram проигрывали их в музыкальном проигрывателе. Ваше аудио должно быть в формате .MP3 или .M4A. В случае успеха, возвращается отправленное сообщение (Message). На текущий момент, боты могут отправлять аудио весом до 50 MB, это ограничение может измениться в будущем.

Для отправки голосовых сообщений, используйте метод send_voice

Документация Telegram: https://core.telegram.org/bots/api#sendaudio

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `audio` (`str` or *telebot.types.InputFile*) – Аудио для отправки. Передайте file_id (String), чтобы отправить аудио, которое уже загружено на сервера Telegram (рекомендуется), передайте HTTP URL (String), чтобы отправить аудио из интернета или загрузите новое с помощью multipart/form-data. Аудио должно быть в формате .MP3 или .M4A.
- `caption` (`str`) – Подпись к аудио, 0-1024 символа после форматирования
- `duration` (`int`) – Длительность аудио в секундах
- `performer` (`str`) – Исполнитель
- `title` (`str`) – Название трека

- reply_to_message_id (int) – Deprecated - Use reply_parameters instead. 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) – Режим форматирования подписи к аудио. См. formatting options для получения подробностей.
- disable_notification (bool) – Отправить сообщение, при получении которого пользователи получат уведомление без звука.
- timeout (int) – Таймаут запроса в секундах.
- thumbnail (str or *telebot.types.InputFile*) – Обложка отправленного файла; может быть проигнорирована, если генерация обложки поддерживается на стороне сервера. Обложка должна быть картинкой в формате JPEG и весить менее 200 kB. Ширина и высота обложки не должны превышать 320. Игнорируется, если файл не загружен с помощью multipart/form-data. Обложки не могут быть использованы повторно и могут быть загружены только как новый файл, так что вы можете передать "attach://<file_attach_name>" если обложка была загружена с помощью multipart/form-data под именем <file_attach_name>.
- caption_entities (list of *telebot.types.MessageEntity*) – Список отформатированных частей подписи в формате JSON, можно использовать вместо parse_mode
- allow_sending_without_reply (bool) – Deprecated - Use reply_parameters instead. Pass True, if the message should be sent even if the specified replied-to message is not found
- protect_content (bool) – Запретить пересылку и сохранение содержимого сообщения
- message_thread_id (int) – id топика, в который нужно отправить сообщение
- thumb (str or *telebot.types.InputFile*) – Deprecated. Use thumbnail instead
- reply_parameters (*telebot.types.ReplyParameters*) – Reply parameters.
- business_connection_id (str) – Unique identifier for the target business connection

Результат

В случае успеха возвращает отправленное сообщение (Message).

Тип результата

telebot.types.Message

`async send_chat_action(`*chat_id: int | str*, *action: str*, *timeout: int | None = None*, *message_thread_id: int | None = None*, *business_connection_id: str | None = None*`)` → bool

Используйте этот метод, когда вам нужно показать пользователю, что бот что-то делает. Статус устанавливается на 5 секунд или менее (когда от бота приходит сообщение, клиенты (приложения) Telegram убирают статус typing). Возвращает True в случае успеха.

Пример: ImageBot-у требуется время, чтобы обработать запрос и загрузить изображение. Вместо отправки текстового сообщения "Отправка изображения, пожалуйста подождите...", бот может использовать sendChatAction с параметром action = upload_photo. Пользователь увидит статус бота "sending photo".

Документация Telegram: https://core.telegram.org/bots/api#sendchataction

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала
- `action` (`str`) – Тип действия. Выберите один, в зависимости от того, что получит пользователь: typing для текстовых сообщений, upload_photo для фото, record_video или upload_video для видео, record_voice или upload_voice для голосовых сообщений, upload_document для файлов, choose_sticker для стикеров, find_location для данных о местоположении, record_video_note или upload_video_note для видео заметок (кружочков).
- `timeout` (`int`) – Таймаут запроса в секундах.
- `message_thread_id` (`int`) – id топика, в который сообщение будет отправлено(только для супергрупп)
- `business_connection_id` (`str`) – Identifier of a business connection, in which the message will be sent

Результат

Возвращает True в случае успеха.

Тип результата

`bool`

`async send_contact(`*chat_id: int | str*, *phone_number: str*, *first_name: str*, *last_name: str | None = None*, *vcard: str | None = None*, *disable_notification: bool | None = None*, *reply_to_message_id: int | None = None*, *reply_markup:* InlineKeyboardMarkup | ReplyKeyboardMarkup | ReplyKeyboardRemove | ForceReply | *None = None*, *timeout: int | None = None*, *allow_sending_without_reply: bool | None = None*, *protect_content: bool | None = None*, *message_thread_id: int | None = None*, *reply_parameters:* ReplyParameters | *None = None*, *business_connection_id: str | None = None*`)` → *Message*

Используйте этот метод, чтобы отправить контакт. В случае успеха, возвращается отправленное сообщение (Message).

Документация Telegram: https://core.telegram.org/bots/api#sendcontact

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала
- `phone_number` (`str`) – Телефонный номер контакта
- `first_name` (`str`) – Имя контакта
- `last_name` (`str`) – Фамилия контакта
- `vcard` (`str`) – Дополнительные данные о контакте в формате vCard, 0-2048 байт
- `disable_notification` (`bool`) – Отправить сообщение, при получении которого пользователи получат уведомление без звука.
- `reply_to_message_id` (`int`) – Deprecated - Use reply_parameters instead. 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`*) – Дополнительные элементы интерфейса. Inline клавиа-

тура, текстовая клавиатура, запрос на удаление текстовой клавиатуры или запрос на ответ от пользователя.

- `timeout` (`int`) – Таймаут запроса в секундах.
- `allow_sending_without_reply` (`bool`) – Deprecated - Use reply_parameters instead. Pass True, if the message should be sent even if one of the specified replied-to messages is not found.
- `protect_content` (`bool`) – Запретить пересылку и сохранение содержимого сообщения
- `message_thread_id` (`int`) – Топик, в который сообщение будет отправлено
- `reply_parameters` (*telebot.types.ReplyParameters*) – Reply parameters.
- `business_connection_id` (`str`) – Identifier of a business connection, in which the message will be sent

Результат

В случае успеха возвращает отправленное сообщение (Message).

Тип результата

telebot.types.Message

`async send_dice(`*chat_id: int | str, emoji: str | None = None, disable_notification: bool | None = None, reply_to_message_id: int | None = None, reply_markup:* InlineKeyboardMarkup | ReplyKeyboardMarkup | ReplyKeyboardRemove | ForceReply *| None = None, timeout: int | None = None, allow_sending_without_reply: bool | None = None, protect_content: bool | None = None, message_thread_id: int | None = None, reply_parameters:* ReplyParameters *| None = None, business_connection_id: str | None = None*`)` → *Message*

Используйте этот метод, чтобы отправить анимированный эмодзи, который покажет случайное значение. В случае успеха, возвращается отправленное сообщение (Message).

Документация Telegram: https://core.telegram.org/bots/api#senddice

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `emoji` (`str`) – Эмодзи, на котором основана анимация. На текущий момент, должно быть одним из “”, “”, “”, “”, “”, или “”. Значение может быть 1-6 для “”, “” и “”, 1-5 для “” и “”, и 1-64 для “”. По умолчанию “”
- `disable_notification` (`bool`) – Отправить сообщение, при получении которого пользователи получат уведомление без звука.
- `reply_to_message_id` (`int`) – Deprecated - Use reply_parameters instead. 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*) – Дополнительные элементы интерфейса. Inline клавиатура, текстовая клавиатура, запрос на удаление текстовой клавиатуры или запрос на ответ от пользователя.
- `timeout` (`int`) – Таймаут запроса в секундах.

- `allow_sending_without_reply` (`bool`) – Deprecated - Use reply_parameters instead. Pass True, if the message should be sent even if the specified replied-to message is not found
- `protect_content` (`bool`) – Защищает содержимое отправленного сообщения от пересылки
- `message_thread_id` (`int`) – id топика, в который сообщение будет отправлено
- `reply_parameters` (*telebot.types.ReplyParameters*) – Reply parameters.
- `business_connection_id` (`str`) – Unique identifier for the target business connection

Результат

В случае успеха возвращает отправленное сообщение (Message).

Тип результата

telebot.types.Message

`async send_document(`*chat_id: int | str, document: Any | str, reply_to_message_id: int | None = None, caption: str | None = None, reply_markup:* InlineKeyboardMarkup | ReplyKeyboardMarkup | ReplyKeyboardRemove | ForceReply *| None = None, parse_mode: str | None = None, disable_notification: bool | None = None, timeout: int | None = None, thumbnail: str | Any | None = None, caption_entities: List[*MessageEntity*] | None = None, allow_sending_without_reply: bool | None = None, visible_file_name: str | None = None, disable_content_type_detection: bool | None = None, data: str | Any | None = None, protect_content: bool | None = None, message_thread_id: int | None = None, thumb: str | Any | None = None, reply_parameters:* ReplyParameters *| None = None, business_connection_id: str | None = None*`) → *Message*`

Используйте этот метод, чтобы отправить файл.

Документация Telegram: https://core.telegram.org/bots/api#senddocument

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `document` (`str` or *telebot.types.InputFile*) – (документ) Файл для отправки. Передайте file_id (String), чтобы отправить файл, который уже загружен на сервера Telegram (рекомендуется), передайте HTTP URL (String), чтобы отправить файл из интернета или загрузите новый с помощью multipart/form-data
- `reply_to_message_id` (`int`) – Deprecated - Use reply_parameters instead. If the message is a reply, ID of the original message
- `caption` (`str`) – Подпись к файлу (может быть использована при повторной отправке файла по file_id), 0-1024 символа после форматирования
- `reply_markup` (*telebot.types.InlineKeyboardMarkup* or *telebot.types.ReplyKeyboardMarkup* or *telebot.types.ReplyKeyboardRemove* or *telebot.types.ForceReply*) – Дополнительные элементы интерфейса. Inline клавиатура, текстовая клавиатура, запрос на удаление текстовой клавиатуры или запрос на ответ от пользователя.
- `parse_mode` (`str`) – Режим форматирования частей подписи к файлу

- `disable_notification` (`bool`) – Отправить сообщение, при получении которого пользователи получат уведомление без звука.
- `timeout` (`int`) – Таймаут запроса в секундах.
- `thumbnail` (`str` or *`telebot.types.InputFile`*) – InputFile или String : Обложка отправленного файла; может быть проигнорирована, если генерация обложки поддерживается на стороне сервера. Обложка должна быть картинкой в формате JPEG и весить менее 200 kB. Ширина и высота обложки не должны превышать 320. Игнорируется, если файл не загружен с помощью multipart/form-data. Обложки не могут быть использованы повторно и могут быть загружены только как новый файл, так что вы можете передать "attach://<file_attach_name>" если обложка была загружена с помощью multipart/form-data под именем <file_attach_name>.
- `caption_entities` (`list` of *`telebot.types.MessageEntity`*) – Список отформатированных частей подписи в формате JSON, можно использовать вместо parse_mode
- `allow_sending_without_reply` (`bool`) – Deprecated - Use reply_parameters instead. Pass True, if the message should be sent even if the specified replied-to message is not found
- `visible_file_name` (`str`) – позволяет задать имя файла, которое будет показано в Telegram вместо настоящего
- `disable_content_type_detection` (`bool`) – Отключает автоматическое обнаружение типа файла на стороне сервера для файлов, загруженных с помощью multipart/form-data
- `data` (`str`) – опечатка: не используйте
- `protect_content` (`bool`) – Запретить пересылку и сохранение содержимого сообщения
- `message_thread_id` (`int`) – id топика, в который нужно отправить сообщение
- `thumb` (`str` or *`telebot.types.InputFile`*) – Deprecated. Use thumbnail instead
- `reply_parameters` (*`telebot.types.ReplyParameters`*) – Reply parameters.
- `business_connection_id` (`str`) – Unique identifier for the target business connection

Результат

В случае успеха возвращает отправленное сообщение (Message).

Тип результата

`telebot.types.Message`

`async send_game(`*chat_id: int | str, game_short_name: str, disable_notification: bool | None = None, reply_to_message_id: int | None = None, reply_markup: InlineKeyboardMarkup | ReplyKeyboardMarkup | ReplyKeyboardRemove | ForceReply | None = None, timeout: int | None = None, allow_sending_without_reply: bool | None = None, protect_content: bool | None = None, message_thread_id: int | None = None, reply_parameters: ReplyParameters | None = None, business_connection_id: str | None = None*`)` → *Message*

Используется для отправки игры.

Документация Telegram: https://core.telegram.org/bots/api#sendgame

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `game_short_name` (`str`) – Короткое имя игры, служит в качестве уникального id игры. Настройте свои игрычерез @BotFather.
- `disable_notification` (`bool`) – Отправить сообщение, при получении которого пользователи получат уведомление без звука.
- `reply_to_message_id` (`int`) – Deprecated - Use reply_parameters instead. If the message is a reply, ID of the original message
- `reply_markup` (`InlineKeyboardMarkup` or `ReplyKeyboardMarkup` or `ReplyKeyboardRemove` or `ForceReply`) – Дополнительные элементы интерфейса. Inline клавиатура, текстовая клавиатура, запрос на удаление текстовой клавиатуры или запрос на ответ от пользователя.
- `timeout` (`int`) – Тайм-аут в секундах, ожидание ответа от бота.
- `allow_sending_without_reply` (`bool`) – Deprecated - Use reply_parameters instead. Pass True, if the message should be sent even if one of the specified replied-to messages is not found.
- `protect_content` (`bool`) – Передайте True, если содержимое сообщение должно быть защищено от просмотра ботом.
- `message_thread_id` (`int`) – id топика, в которые будет сообщение будет отправлено.
- `reply_parameters` (*`telebot.types.ReplyParameters`*) – Reply parameters.
- `business_connection_id` (`str`) – Identifier of the business connection.

Результат

В случае успеха возвращает отправленное сообщение (Message).

Тип результата

`types.Message`

```
async send_invoice(chat_id: int | str, title: str, description: str, invoice_payload: str,
                   provider_token: str, currency: str, prices: List[LabeledPrice],
                   start_parameter: str | None = None, photo_url: str | None = None,
                   photo_size: int | None = None, photo_width: int | None = None,
                   photo_height: int | None = None, need_name: bool | None = None,
                   need_phone_number: bool | None = None, need_email: bool | None = None,
                   need_shipping_address: bool | None = None,
                   send_phone_number_to_provider: bool | None = None,
                   send_email_to_provider: bool | None = None, is_flexible: bool | None =
                   None, disable_notification: bool | None = None, reply_to_message_id: int |
                   None = None, reply_markup: InlineKeyboardMarkup |
                   ReplyKeyboardMarkup | ReplyKeyboardRemove | ForceReply | None =
                   None, provider_data: str | None = None, timeout: int | None = None,
                   allow_sending_without_reply: bool | None = None, max_tip_amount: int |
                   None = None, suggested_tip_amounts: List[int] | None = None,
                   protect_content: bool | None = None, message_thread_id: int | None = None,
                   reply_parameters: ReplyParameters | None = None) → Message
```

Отправляет инвойс.

Документация Telegram: https://core.telegram.org/bots/api#sendinvoice

Параметры

- `chat_id` (`int` or `str`) – Уникальный id приватного чата
- `title` (`str`) – Название товара, 1-32 символа
- `description` (`str`) – Описание товара, 1-255 символов
- `invoice_payload` (`str`) – Дополнительные данные, 1-128 байт. Не будет показано пользователю, используйте во внутренних процессах.
- `provider_token` (`str`) – Токен платежной системы, полученный через @BotFather
- `currency` (`str`) – Трехбуквенный код валюты в формате ISO 4217, см. https://core.telegram.org/bots/payments#supported-currencies
- `prices` (List[`types.LabeledPrice`]) – Детали цены, список компонент (например цена продукта, налог, скидка, стоимость доставки, налог на доставку, бонус и т.д.)
- `start_parameter` (`str`) – Уникальный deep-linking параметр, который может быть использован для генерации этого инвойса при использовании в качестве параметра /start
- `photo_url` (`str`) – URL фото продукта. Может быть фото товаров или рекламным изображением сервиса. Людям больше нравится, когда они видят, за что платят.
- `photo_size` (`int`) – Вес изображения в байтах
- `photo_width` (`int`) – Ширина изображения
- `photo_height` (`int`) – Высота изображения
- `need_name` (`bool`) – Передайте True, если для совершения заказа требуется полное имя пользователя
- `need_phone_number` (`bool`) – Передайте True, если для совершения заказа требуется номер телефона пользователя
- `need_email` (`bool`) – Передайте True, если для совершения заказа требуется email пользователя
- `need_shipping_address` (`bool`) – Передайте True, если для совершения заказа требуется адрес доставки
- `is_flexible` (`bool`) – Передайте True, если окончательная цена зависит от способа доставки
- `send_phone_number_to_provider` (`bool`) – Передайте True, если номер телефона пользователя нужно отправить платежной системе
- `send_email_to_provider` (`bool`) – Передайте True, если email пользователя нужно отправить платежной системе
- `disable_notification` (`bool`) – Отправить сообщение, при получении которого пользователи получат уведомление без звука.
- `reply_to_message_id` (`int`) – Deprecated - Use reply_parameters instead. If the message is a reply, ID of the original message
- `reply_markup` (`str`) – JSON-сериализованный объект inline клавиатуры. Если пустой, будет показана одна кнопка „Pay total price“. Если не пустой, первая кнопка должна быть кнопкой для оплаты

- `provider_data` (`str`) – Данные о инвойсе в формате JSON, которые будут переданы платежной системе. Подробное описание обязательных полей должно быть предоставлено провайдером платежной системы.
- `timeout` (`int`) – Тайм-аут запроса, по умолчанию None
- `allow_sending_without_reply` (`bool`) – Deprecated - Use reply_parameters instead. Pass True, if the message should be sent even if the specified replied-to message is not found
- `max_tip_amount` (`int`) – Максимальный размер чаевых в наименьших единицах выбранной валюты
- `suggested_tip_amounts` (`list` of `int`) – Массив предлагаемых вариантов чаевых в наименьших единицах выбранной валюты в формате JSON. Можно задать не более 4 вариантов. Варианты чаевых должны быть больше нуля, перечисленные в порядке строгого возрастания и не превышать max_tip_amount.
- `protect_content` (`bool`) – Запретить пересылку и сохранение содержимого сообщения
- `message_thread_id` (`int`) – id топика, в который будет отправлен инвойс
- `reply_parameters` (*`telebot.types.ReplyParameters`*) – Reply parameters.

Результат

В случае успеха возвращает отправленное сообщение (Message).

Тип результата

`types.Message`

`async send_location(`*chat_id: int | str, latitude: float, longitude: float, live_period: int | None = None, reply_to_message_id: int | None = None, reply_markup:* InlineKeyboardMarkup | ReplyKeyboardMarkup | ReplyKeyboardRemove | ForceReply *| None = None, disable_notification: bool | None = None, timeout: int | None = None, horizontal_accuracy: float | None = None, heading: int | None = None, proximity_alert_radius: int | None = None, allow_sending_without_reply: bool | None = None, protect_content: bool | None = None, message_thread_id: int | None = None, reply_parameters:* ReplyParameters *| None = None, business_connection_id: str | None = None*`) →` *Message*

Используйте этот метод, чтобы отправить точку на карте. В случае успеха, возвращается отправленное сообщение (Message).

Документация Telegram: https://core.telegram.org/bots/api#sendlocation

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `latitude` (`float`) – Широта
- `longitude` (`float`) – Долгота
- `live_period` (`int`) – Время в секундах, в течение которого местоположение будет обновляться (см. Live Locations), должно быть между 60 и 86400.
- `reply_to_message_id` (`int`) – Deprecated - Use reply_parameters instead. 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*) – Дополнительные элементы интерфейса. Inline клавиатура, текстовая клавиатура, запрос на удаление текстовой клавиатуры или запрос на ответ от пользователя.
- disable_notification (bool) – Отправить сообщение, при получении которого пользователи получат уведомление без звука.
- timeout (int) – Таймаут запроса в секундах.
- horizontal_accuracy (float) – Радиус погрешности местоположения, измеряется в метрах; 0-1500
- heading (int) – Для live местоположений, направление, в котором пользователь двигается, в градусах. Должно быть между 1 и 360, если указано.
- proximity_alert_radius (int) – Для live местоположений, максимальное расстояние для уведомлений о приближении другого участника чата, в метрах. Должно быть между 1 и 100000, если указано.
- allow_sending_without_reply (bool) – Deprecated - Use reply_parameters instead. Pass True, if the message should be sent even if the specified replied-to message is not found
- protect_content (bool) – Запретить пересылку и сохранение содержимого сообщения
- message_thread_id (int) – id топика, в который нужно отправить сообщение
- reply_parameters (*telebot.types.ReplyParameters*) – Reply parameters.
- business_connection_id (str) – Identifier of a business connection, in which the message will be sent

Результат

В случае успеха возвращает отправленное сообщение (Message).

Тип результата

telebot.types.Message

async send_media_group(*chat_id: int | str*, *media: List[InputMediaAudio | InputMediaDocument | InputMediaPhoto | InputMediaVideo]*, *disable_notification: bool | None = None*, *protect_content: bool | None = None*, *reply_to_message_id: int | None = None*, *timeout: int | None = None*, *allow_sending_without_reply: bool | None = None*, *message_thread_id: int | None = None*, *reply_parameters: ReplyParameters | None = None*, *business_connection_id: str | None = None*) → List[*Message*]

Используйте этот метод, чтобы отправить группу фото, видео, файлов или аудио как альбом. Файлы и аудио могут быть сгруппированы в альбом только с сообщениями того же типа. В случае успеха, возвращается массив отправленных сообщений (Message).

Документация Telegram: https://core.telegram.org/bots/api#sendmediagroup

Параметры

- chat_id (int or str) – Уникальный id чата или username канала (в формате @channelusername)
- media (list of types.InputMedia) – JSON-сериализованный массив, описывающий сообщения для отправки, должен включать от 2 до 10 элементов

- `disable_notification` (`bool`) – Отправить сообщение, при получении которого пользователя пользователи получат уведомление без звука.
- `protect_content` (`bool`) – Запретить пересылку и сохранение содержимого сообщения
- `reply_to_message_id` (`int`) – Deprecated - Use reply_parameters instead. If the message is a reply, ID of the original message
- `timeout` (`int`) – Таймаут запроса в секундах.
- `allow_sending_without_reply` (`bool`) – Deprecated - Use reply_parameters instead. Pass True, if the message should be sent even if the specified replied-to message is not found
- `message_thread_id` (`int`) – id топика, в который будет отправлена группа медиа
- `reply_parameters` (*`telebot.types.ReplyParameters`*) – Reply parameters.
- `business_connection_id` (`str`) – Identifier of a business connection, in which the message will be sent

Результат

В случае успеха, возвращается массив отправленных сообщений (Message).

Тип результата

List[*types.Message*]

`async send_message(`*chat_id: int | str, text: str, parse_mode: str | None = None, entities: List[MessageEntity] | None = None, disable_web_page_preview: bool | None = None, disable_notification: bool | None = None, protect_content: bool | None = None, reply_to_message_id: int | None = None, allow_sending_without_reply: bool | None = None, reply_markup: InlineKeyboardMarkup | ReplyKeyboardMarkup | ReplyKeyboardRemove | ForceReply | None = None, timeout: int | None = None, message_thread_id: int | None = None, reply_parameters: ReplyParameters | None = None, link_preview_options: LinkPreviewOptions | None = None, business_connection_id: str | None = None*`) →` *Message*

Используйте этот метод, чтобы отправлять текстовые сообщения.

Предупреждение: Не отправляйте больше 4096 символов в одном сообщении, иначе вы рискуете получить ошибку HTTP 414. Если вам нужно отправить больше 4096 символов, используйте функцию *split_string* или *smart_split* из util.py.

Документация Telegram: https://core.telegram.org/bots/api#sendmessage

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `text` (`str`) – Текст сообщения для отправки
- `parse_mode` (`str`) – Режим форматирования в тексте сообщения.
- `entities` (Array of *`telebot.types.MessageEntity`*) – Список отформатированных частей в тексте сообщения, можно использовать вместо parse_mode
- `disable_web_page_preview` (`bool`) – Deprecated - Use link_preview_options instead.

- `disable_notification` (`bool`) – Отправить сообщение, при получении которого пользователи получат уведомление без звука.
- `protect_content` (`bool`) – Если True, содержимое сообщения будет скрыто от всех пользователей, кроме заданного
- `reply_to_message_id` (`int`) – Deprecated - Use reply_parameters instead. If the message is a reply, ID of the original message
- `allow_sending_without_reply` (`bool`) – Deprecated - Use reply_parameters instead. 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*) – Дополнительные элементы интерфейса. Inline клавиатура, текстовая клавиатура, запрос на удаление текстовой клавиатуры или запрос на ответ от пользователя.
- `timeout` (`int`) – Таймаут запроса в секундах.
- `message_thread_id` (`int`) – Уникальный id топика, в который нужно переслать сообщение; только для супергрупп с топиками
- `reply_parameters` (*telebot.types.ReplyParameters*) – Reply parameters.
- `link_preview_options` (*telebot.types.LinkPreviewOptions*) – Options for previewing links.
- `business_connection_id` (`str`) – Unique identifier for the target business connection

Результат

В случае успеха возвращает отправленное сообщение (Message).

Тип результата

telebot.types.Message

`async send_photo(`*chat_id: int | str, photo: Any | str, caption: str | None = None, parse_mode: str | None = None, caption_entities: List[MessageEntity] | None = None, disable_notification: bool | None = None, protect_content: bool | None = None, reply_to_message_id: int | None = None, allow_sending_without_reply: bool | None = None, reply_markup: InlineKeyboardMarkup | ReplyKeyboardMarkup | ReplyKeyboardRemove | ForceReply | None = None, timeout: int | None = None, message_thread_id: int | None = None, has_spoiler: bool | None = None, reply_parameters: ReplyParameters | None = None, business_connection_id: str | None = None*`) → *Message*

Используйте этот метод, чтобы отправить фото. В случае успеха, возвращается отправленное сообщение (Message).

Документация Telegram: https://core.telegram.org/bots/api#sendphoto

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `photo` (`str` or *telebot.types.InputFile*) – Фото для отправки. Передайте file_id (String), чтобы отправить фото, которое уже загружено на сервера Telegram (рекомендуется), передайте HTTP URL (String), чтобы отправить фото из интернета или загрузите новое с помощью multipart/form-data. Фото

должно весить не более 10 MB. Ширина и высота фото не должны суммарно превышать 10000. Отношение ширины и высоты должно быть не более 20.

- `caption` (`str`) – Подпись к фото (может быть использована при повторной отправке файла по file_id), 0-1024 символа после форматирования
- `parse_mode` (`str`) – Режим форматирования подписи к фото.
- `caption_entities` (`list of` *`telebot.types.MessageEntity`*) – Список отформатированных частей подписи в формате JSON, можно использовать вместо parse_mode
- `disable_notification` (`bool`) – Отправить сообщение, при получении которого пользователи получат уведомление без звука.
- `protect_content` (`bool`) – Запретить пересылку и сохранение содержимого сообщения
- `reply_to_message_id` (`int`) – Deprecated - Use reply_parameters instead. If the message is a reply, ID of the original message
- `allow_sending_without_reply` (`bool`) – Deprecated - Use reply_parameters instead. 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`*) – Дополнительные элементы интерфейса. Inline клавиатура, текстовая клавиатура, запрос на удаление текстовой клавиатуры или запрос на ответ от пользователя.
- `timeout` (`int`) – Таймаут запроса в секундах.
- `message_thread_id` (`int`) – id топика, в который нужно отправить сообщение
- `has_spoiler` (`bool`) – Передайте True, если фото должно быть отправлено как спойлер
- `reply_parameters` (*`telebot.types.ReplyParameters`*) – Reply parameters.
- `business_connection_id` (`str`) – Unique identifier for the target business connection

Результат

В случае успеха возвращает отправленное сообщение (Message).

Тип результата

`telebot.types.Message`

`async send_poll(`*chat_id: int | str, question: str, options: List[str], is_anonymous: bool | None = None, type: str | None = None, allows_multiple_answers: bool | None = None, correct_option_id: int | None = None, explanation: str | None = None, explanation_parse_mode: str | None = None, open_period: int | None = None, close_date: int | datetime | None = None, is_closed: bool | None = None, disable_notification: bool | None = False, reply_to_message_id: int | None = None, reply_markup:* InlineKeyboardMarkup | ReplyKeyboardMarkup | ReplyKeyboardRemove | ForceReply | *None = None, allow_sending_without_reply: bool | None = None, timeout: int | None = None, explanation_entities: List[*MessageEntity*] | None = None, protect_content: bool | None = None, message_thread_id: int | None = None, reply_parameters:* ReplyParameters | *None = None, business_connection_id: str | None = None*`)`
→ *Message*

Используйте этот метод, чтобы отправить опрос. В случае успеха, возвращается отправленное сообщение (Message).

Документация Telegram: https://core.telegram.org/bots/api#sendpoll

Параметры

- `chat_id` (`int` | `str`) – Уникальный id чата или username канала
- `question` (`str`) – Тема опроса, 1-300 символов
- `options` (`list` of `str`) – JSON-сериализованный список вариантов ответа, 2-10 строк по 1-100 символов
- `is_anonymous` (`bool`) – True, если опрос должен быть анонимным, по умолчанию True
- `type` (`str`) – Вид опроса, "quiz" или "regular", по умолчанию "regular"
- `allows_multiple_answers` (`bool`) – True, если опрос позволяет выбрать несколько вариантов ответа, игнорируется в опросах вида "quiz", по умолчанию False
- `correct_option_id` (`int`) – Индекс правильного варианта ответа, начиная с 0. Доступно только для опросов вида "quiz", по умолчанию None
- `explanation` (`str`) – Текст, который будет показан при выборе неправильно варианта ответа или нажатии на иконку лампочки в опросах вида "quiz", 0-200 символов и не более 2 строк после форматирования
- `explanation_parse_mode` (`str`) – Режим форматирования explanation. См. formatting options для получения подробностей.
- `open_period` (`int`) – Время в секундах, в течение которого опрос будет активен, 5-600. Нельзя использовать вместо с close_date.
- `close_date` (`int` | `datetime`) – Время (UNIX timestamp), когда опрос будет автоматически завершен.
- `is_closed` (`bool`) – Передайте True, если опрос должен быть завершен немедленно. Может быть полезно для предпросмотра опроса.
- `disable_notification` (`bool`) – Отправить сообщение, при получении которого пользователи получат уведомление без звука.
- `reply_to_message_id` (`int`) – Deprecated - Use reply_parameters instead. If the message is a reply, ID of the original message
- `allow_sending_without_reply` (`bool`) – Deprecated - Use reply_parameters instead. Pass True, if the poll allows multiple options to be voted simultaneously.
- `reply_markup` (`InlineKeyboardMarkup` | `ReplyKeyboardMarkup` | `ReplyKeyboardRemove` | `ForceReply`) – Дополнительные элементы интерфейса. Inline клавиатура, текстовая клавиатура, запрос на удаление текстовой клавиатуры или запрос на ответ от пользователя.
- `timeout` (`int`) – Тайм-аут в секундах, ожидание ответа от пользователя.
- `explanation_entities` (`list` of `MessageEntity`) – JSON-сериализованный список отформатированных частей explanation, можно использовать вместо parse_mode

- `protect_content` (`bool`) – Запретить пересылку и сохранение содержимого сообщения
- `message_thread_id` (`int`) – id топика, в который будет отправлен опрос
- `reply_parameters` (*`telebot.types.ReplyParameters`*) – Reply parameters.
- `business_connection_id` (`str`) – Identifier of the business connection to send the message through

Результат
В случае успеха возвращает отправленное сообщение (Message).

Тип результата
`types.Message`

`async send_sticker(`*chat_id: int | str, sticker: Any | str, reply_to_message_id: int | None = None, reply_markup:* InlineKeyboardMarkup | ReplyKeyboardMarkup | ReplyKeyboardRemove | ForceReply | *None = None, disable_notification: bool | None = None, timeout: int | None = None, allow_sending_without_reply: bool | None = None, protect_content: bool | None = None, data: Any | str = None, message_thread_id: int | None = None, emoji: str | None = None, reply_parameters:* ReplyParameters | *None = None, business_connection_id: str | None = None*`)` → *Message*

Используйте этот метод, чтобы отправить статичный .WEBP, анимированный .TGS, или видео .WEBM стикер. В случае успеха возвращает отправленное сообщение (Message).

Документация Telegram: https://core.telegram.org/bots/api#sendsticker

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `sticker` (`str` or *`telebot.types.InputFile`*) – Стикер для отправки. Передайте file_id (String), чтобы отправить файл, который уже загружен на сервера Telegram (рекомендуется), передайте HTTP URL (String), чтобы отправить .webp файл из интернета или загрузите новый с помощью multipart/form-data.
- `reply_to_message_id` (`int`) – Deprecated - Use reply_parameters instead. 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`*) – Дополнительные элементы интерфейса. Inline клавиатура, текстовая клавиатура, запрос на удаление текстовой клавиатуры или запрос на ответ от пользователя.
- `disable_notification` (`bool`) – отключить уведомление
- `timeout` (`int`) – Таймаут запроса в секундах.
- `allow_sending_without_reply` (`bool`) – Deprecated - Use reply_parameters instead. Pass True, if the message should be sent even if the specified replied-to message is not found
- `protect_content` (`bool`) – Запретить пересылку и сохранение содержимого сообщения
- `data` (`str`) – опечатка: не используйте
- `message_thread_id` (`int`) – id топика, в который нужно отправить сообщение

- emoji (str) – Emoji associated with the sticker; only for just uploaded stickers
- reply_parameters (*telebot.types.ReplyParameters*) – Reply parameters.
- business_connection_id (str) – Unique identifier for the target business connection

Результат
В случае успеха возвращает отправленное сообщение (Message).

Тип результата
telebot.types.Message

async send_venue(*chat_id: int | str*, *latitude: float*, *longitude: float*, *title: str*, *address: str*, *foursquare_id: str | None = None*, *foursquare_type: str | None = None*, *disable_notification: bool | None = None*, *reply_to_message_id: int | None = None*, *reply_markup:* InlineKeyboardMarkup | ReplyKeyboardMarkup | ReplyKeyboardRemove | ForceReply | *None = None*, *timeout: int | None = None*, *allow_sending_without_reply: bool | None = None*, *google_place_id: str | None = None*, *google_place_type: str | None = None*, *protect_content: bool | None = None*, *message_thread_id: int | None = None*, *reply_parameters:* ReplyParameters | *None = None*, *business_connection_id: str | None = None*) → *Message*

Используйте этот метод, чтобы отправить информацию о месте. В случае успеха возвращается отправленное сообщение (Message).

Документация Telegram: https://core.telegram.org/bots/api#sendvenue

Параметры

- chat_id (int or str) – Уникальный id чата или username канала
- latitude (float) – Широта
- longitude (float) – Долгота
- title (str) – Название места
- address (str) – Адрес места
- foursquare_id (str) – id места на Foursquare
- foursquare_type (str) – Тип места на Foursquare, если известен. (Например, “arts_entertainment/default”, “arts_entertainment/aquarium” или “food/icecream”.)
- disable_notification (bool) – Отправить сообщение, при получении которого пользователи получат уведомление без звука.
- reply_to_message_id (int) – Deprecated - Use reply_parameters instead. 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*) – Дополнительные элементы интерфейса. Inline клавиатура, текстовая клавиатура, запрос на удаление текстовой клавиатуры или запрос на ответ от пользователя.
- timeout (int) – Таймаут запроса в секундах.
- allow_sending_without_reply (bool) – Deprecated - Use reply_parameters instead. Pass True, if the message should be sent even if one of the specified replied-to messages is not found.

- `google_place_id` (`str`) – id места на Google Places
- `google_place_type` (`str`) – Тип места на Google Places.
- `protect_content` (`bool`) – Запретить пересылку и сохранение содержимого сообщения
- `message_thread_id` (`int`) – Топик, в который сообщение будет отправлено
- `reply_parameters` (*telebot.types.ReplyParameters*) – Reply parameters.
- `business_connection_id` (`str`) – Identifier of a business connection, in which the message will be sent

Результат
В случае успеха возвращает отправленное сообщение (Message).

Тип результата
telebot.types.Message

`async send_video(`*chat_id: int | str, video: Any | str, duration: int | None = None, width: int | None = None, height: int | None = None, thumbnail: str | Any | None = None, caption: str | None = None, parse_mode: str | None = None, caption_entities: List[MessageEntity] | None = None, supports_streaming: bool | None = None, disable_notification: bool | None = None, protect_content: bool | None = None, reply_to_message_id: int | None = None, allow_sending_without_reply: bool | None = None, reply_markup: InlineKeyboardMarkup | ReplyKeyboardMarkup | ReplyKeyboardRemove | ForceReply | None = None, timeout: int | None = None, data: str | Any | None = None, message_thread_id: int | None = None, has_spoiler: bool | None = None, thumb: str | Any | None = None, reply_parameters: ReplyParameters | None = None, business_connection_id: str | None = None*`) → `Message`

Используйте этот метод, чтобы отправить видео, клиенты (приложения) Telegram поддерживают mp4 видео (другие форматы могут быть отправлены как Document).

Документация Telegram: https://core.telegram.org/bots/api#sendvideo

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `video` (`str` or *telebot.types.InputFile*) – Видео для отправки. Передайте file_id (String), чтобы отправить видео, которое уже загружено на сервера Telegram или загрузите новое с помощью multipart/form-data.
- `duration` (`int`) – Длительность отправленного видео в секундах
- `width` (`int`) – Ширина видео
- `height` (`int`) – Высота видео
- `thumbnail` (`str` or *telebot.types.InputFile*) – Обложка отправленного файла; может быть проигнорирована, если генерация обложки поддерживается на стороне сервера. Обложка должна быть картинкой в формате JPEG и весить менее 200 kB. Ширина и высота обложки не должны превышать 320. Игнорируется, если файл не загружен с помощью multipart/form-data. Обложки не могут быть использованы повторно и могут быть загружены только как новый файл, так что вы можете передать “attach://<file_attach_name>” если обложка была загружена с помощью multipart/form-data под именем <file_attach_name>.

- caption (str) – Подпись к видео (может быть использована при повторной отправке файла по file_id), 0-1024 символа после форматирования
- parse_mode (str) – Режим форматирования подписи к видео
- caption_entities (list of *telebot.types.MessageEntity*) – Список отформатированных частей подписи, можно использовать вместо parse_mode
- supports_streaming (bool) – Передайте True, если загруженное видео подходит для стриминга
- disable_notification (bool) – Отправить сообщение, при получении которого пользователи получат уведомление без звука.
- protect_content (bool) – Запретить пересылку и сохранение содержимого сообщения
- reply_to_message_id (int) – Deprecated - Use reply_parameters instead. If the message is a reply, ID of the original message
- allow_sending_without_reply (bool) – Deprecated - Use reply_parameters instead. 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*) – Дополнительные элементы интерфейса. Inline клавиатура, текстовая клавиатура, запрос на удаление текстовой клавиатуры или запрос на ответ от пользователя.
- timeout (int) – Таймаут запроса в секундах.
- data (str) – опечатка: не используйте
- message_thread_id (int) – id топика, в который будет отправлено видео
- has_spoiler (bool) – Передайте True, если видео должно быть отправлено как спойлер
- thumb (str or *telebot.types.InputFile*) – Deprecated. Use thumbnail instead
- reply_parameters (*telebot.types.ReplyParameters*) – Reply parameters.
- business_connection_id (str) – Identifier of a business connection, in which the message will be sent

Результат

В случае успеха возвращает отправленное сообщение (Message).

Тип результата

telebot.types.Message

```
async send_video_note(chat_id: int | str, data: Any | str, duration: int | None = None, length: int
                      | None = None, reply_to_message_id: int | None = None, reply_markup:
                      InlineKeyboardMarkup | ReplyKeyboardMarkup | ReplyKeyboardRemove
                      | ForceReply | None = None, disable_notification: bool | None = None,
                      timeout: int | None = None, thumbnail: str | Any | None = None,
                      allow_sending_without_reply: bool | None = None, protect_content: bool |
                      None = None, message_thread_id: int | None = None, thumb: str | Any |
                      None = None, reply_parameters: ReplyParameters | None = None,
                      business_connection_id: str | None = None) → Message
```

Начиная с версии v.4.0, клиенты(приложения) Telegram поддерживают скругленные квадратные MPEG4 видео длительностью до минуты. Используйте этот метод, чтобы отправить видео заметку (кружочек). В случае успеха возвращается отправленное сообщение (Message).

Документация Telegram: https://core.telegram.org/bots/api#sendvideonote

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `data` (`str` or *`telebot.types.InputFile`*) – Видео заметка для отправки. Передайте file_id (String), чтобы отправить видео заметку, которая уже загружена на сервера Telegram или загрузите новую с помощью multipart/form-data. На текущий момент, отправка видео заметок по URL не поддерживается
- `duration` (`int`) – Длительность отправленного видео в секундах
- `length` (`int`) – Ширина и высота видео (диаметр видео сообщения)
- `reply_to_message_id` (`int`) – Deprecated - Use reply_parameters instead. 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`*) – Дополнительные элементы интерфейса. Inline клавиатура, текстовая клавиатура, запрос на удаление текстовой клавиатуры или запрос на ответ от пользователя.
- `disable_notification` (`bool`) – Отправить сообщение, при получении которого пользователи получат уведомление без звука.
- `timeout` (`int`) – Таймаут запроса в секундах.
- `thumbnail` (`str` or *`telebot.types.InputFile`*) – Обложка отправленного файла; может быть проигнорирована, если генерация обложки поддерживается на стороне сервера. Обложка должна быть картинкой в формате JPEG и весить менее 200 kB. Ширина и высота обложки не должны превышать 320. Игнорируется, если файл не загружен с помощью multipart/form-data. Обложки не могут быть использованы повторно и могут быть загружены только как новый файл, так что вы можете передать "attach://<file_attach_name>" если обложка была загружена с помощью multipart/form-data под именем <file_attach_name>.
- `allow_sending_without_reply` (`bool`) – Deprecated - Use reply_parameters instead. Pass True, if the message should be sent even if the specified replied-to message is not found
- `protect_content` (`bool`) – Запретить пересылку и сохранение содержимого сообщения
- `message_thread_id` (`int`) – id топика, в который будет отправлена видео заметка
- `thumb` (`str` or *`telebot.types.InputFile`*) – Deprecated. Use thumbnail instead
- `reply_parameters` (*`telebot.types.ReplyParameters`*) – Reply parameters.
- `business_connection_id` (`str`) – Identifier of a business connection, in which the message will be sent

Результат
В случае успеха возвращает отправленное сообщение (Message).

Тип результата
telebot.types.Message

`async send_voice(chat_id: int | str, voice: Any | str, caption: str | None = None, duration: int | None = None, reply_to_message_id: int | None = None, reply_markup: InlineKeyboardMarkup | ReplyKeyboardMarkup | ReplyKeyboardRemove | ForceReply | None = None, parse_mode: str | None = None, disable_notification: bool | None = None, timeout: int | None = None, caption_entities: List[MessageEntity] | None = None, allow_sending_without_reply: bool | None = None, protect_content: bool | None = None, message_thread_id: int | None = None, reply_parameters: ReplyParameters | None = None, business_connection_id: str | None = None) → Message`

Используйте этот метод, чтобы отправить голосовое сообщение. Ваше аудио должно быть в формате .OGG и закодировано с помощью OPUS (другие форматы можно отправить как Audio или Document). В случае успеха возвращается отправленное сообщение (Message). На текущий момент, боты могут отправлять голосовые сообщения весом до 50 MB, это ограничение может быть изменено в будущем.

Документация Telegram: https://core.telegram.org/bots/api#sendvoice

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `voice` (`str` or *telebot.types.InputFile*) – Аудио для отправки. Передайте file_id (String), чтобы отправить аудио, которое уже загружено на сервера Telegram (рекомендуется), передайте HTTP URL (String), чтобы отправить аудио из интернета или загрузите новое с помощью multipart/form-data.
- `caption` (`str`) – Подпись к голосовому сообщению, 0-1024 символа после форматирования
- `duration` (`int`) – Длительность голосового сообщения в секундах
- `reply_to_message_id` (`int`) – Deprecated - Use reply_parameters instead. 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*) – Дополнительные элементы интерфейса. Inline клавиатура, текстовая клавиатура, запрос на удаление текстовой клавиатуры или запрос на ответ от пользователя.
- `parse_mode` (`str`) – Режим форматирования подписи к голосовому сообщению. См. formatting options для получения подробностей.
- `disable_notification` (`bool`) – Отправить сообщение, при получении которого пользователи получат уведомление без звука.
- `timeout` (`int`) – Таймаут запроса в секундах.
- `caption_entities` (`list` of *telebot.types.MessageEntity*) – Список отформатированных частей подписи в формате JSON, можно использовать вместо parse_mode

- `allow_sending_without_reply` (`bool`) – Deprecated - Use reply_parameters instead. Pass True, if the message should be sent even if the specified replied-to message is not found
- `protect_content` (`bool`) – Запретить пересылку и сохранение содержимого сообщения
- `message_thread_id` (`int`) – id топика, в который нужно отправить сообщение
- `reply_parameters` (*telebot.types.ReplyParameters*) – Reply parameters.
- `business_connection_id` (`str`) – Unique identifier for the target business connection

Результат
В случае успеха возвращает отправленное сообщение (Message).

`async set_chat_administrator_custom_title(`*chat_id: int | str*, *user_id: int*, *custom_title: str*`)` → bool

Используйте этот метод, чтобы задать кастомное звание администратора супергруппы, повышенного ботом. Возвращает True в случае успеха.

Документация Telegram: https://core.telegram.org/bots/api#setchatadministratorcustomtitle

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username супергруппы (в формате @supergroupusername)
- `user_id` (`int`) – Уникальный id сделавшего запрос пользователя
- `custom_title` (`str`) – Новое кастомное звание администратора; 0-16 символов, эмодзи не разрешены

Результат
True в случае успеха.

Тип результата
`bool`

`async set_chat_description(`*chat_id: int | str*, *description: str | None = None*`)` → bool

Используйте этот метод, чтобы изменить описание супергруппы или канала. Бот должен быть администратором чата и иметь соответствующие права администратора.

Документация Telegram: https://core.telegram.org/bots/api#setchatdescription

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `description` (`str`) – Str: Новое описание чата, 0-255 символов

Результат
True в случае успеха.

Тип результата
`bool`

`async set_chat_menu_button(`*chat_id: int | str = None*, *menu_button:* MenuButton *= None*`)` → bool

Используйте этот метод, чтобы изменить кнопку меню в приватном чате или кнопку меню по умолчанию. Возвращает True в случае успеха.

Документация Telegram: https://core.telegram.org/bots/api#setchatmenubutton

Параметры

- `chat_id` (`int` or `str`) – Уникальный id приватного чата. Если не указан, будет изменена кнопка меню по умолчанию.
- `menu_button` (*`telebot.types.MenuButton`*) – JSON-сериализованный объект новой кнопки меню. По умолчанию MenuButtonDefault

Результат
True в случае успеха.

Тип результата
`bool`

`async set_chat_permissions(`*chat_id: int | str*, *permissions:* ChatPermissions, *use_independent_chat_permissions: bool | None = None*`)` → bool

Используйте этот метод, чтобы задать права по умолчанию для всех участников чата. Бот должен быть администратором группы или супергруппы и иметь права администратора can_restrict_members.

Документация Telegram: https://core.telegram.org/bots/api#setchatpermissions

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username супергруппы (в формате @supergroupusername)
- `permissions` (`telebot.types..ChatPermissions`) – Новые права по умолчанию
- `use_independent_chat_permissions` (`bool`) – Pass True if chat permissions are set independently. Otherwise, the can_send_other_messages and can_add_web_page_previews permissions will imply the can_send_messages, can_send_audios, can_send_documents, can_send_photos, can_send_videos, can_send_video_notes, and can_send_voice_notes permissions; the can_send_polls permission will imply the can_send_messages permission.

Результат
True в случае успеха

Тип результата
`bool`

`async set_chat_photo(`*chat_id: int | str*, *photo: Any*`)` → bool

Используйте этот метод, чтобы задать новую аватарку чата. В приватных чатах аватарки менять нельзя. Бот должен быть администратором чата и иметь соответствующие права администратора. Возвращает True в случае успеха. Примечание: В обычных группах (не супергруппы), этот метод будет работать только если настройка 'All Members Are Admins' отключена.

Документация Telegram: https://core.telegram.org/bots/api#setchatphoto

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `photo` (`typing.Union[file_like, str]`) – InputFile: Новая аватарка чата, загруженная с помощью multipart/form-data

Результат
True в случае успеха.

Тип результата
`bool`

`async set_chat_sticker_set(chat_id: int | str, sticker_set_name: str) → StickerSet`

Используйте этот метод, чтобы задать стикерпак супергруппы. Бот должен быть администратором чата и иметь соответствующие права администратора. Используйте атрибут can_set_sticker_set, возвращаемые методом getChat, чтобы проверить, что бот может использовать этот метод. Возвращает True в случае успеха.

Документация Telegram: https://core.telegram.org/bots/api#setchatstickerset

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username супергруппы (в формате @supergroupusername)
- `sticker_set_name` (`str`) – Имя стикерпака для установки в качестве стикерпака группы

Результат
Объект StickerSet

Тип результата
`telebot.types.StickerSet`

`async set_chat_title(chat_id: int | str, title: str) → bool`

Используйте этот метод, чтобы изменить название чата. В приватных чатах изменить название нельзя. Бот должен быть администратором чата и иметь соответствующие права админа. Возвращает True в случае успеха. Примечание: В обычных группах (не супергруппы), этот метод будет работать только если настройка 'All Members Are Admins' отключена.

Документация Telegram: https://core.telegram.org/bots/api#setchattitle

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `title` (`str`) – Новое название чата, 1-255 символов

Результат
True в случае успеха.

Тип результата
`bool`

`async set_custom_emoji_sticker_set_thumbnail(name: str, custom_emoji_id: str | None = None) → bool`

Use this method to set the thumbnail of a custom emoji sticker set. Returns True on success.

Параметры

- `name` (`str`) – Имя стикерпака
- `custom_emoji_id` (`str`) – Custom emoji identifier of a sticker from the sticker set; pass an empty string to drop the thumbnail and use the first sticker as the thumbnail.

Результат
Возвращает True в случае успеха.

Тип результата
`bool`

`async set_game_score(`*user_id: int | str*, *score: int*, *force: bool | None = None*, *chat_id: int | str | None = None*, *message_id: int | None = None*, *inline_message_id: str | None = None*, *disable_edit_message: bool | None = None*`) →` *Message* | bool

Задаёт количество очков пользователя в игре.

Документация Telegram: https://core.telegram.org/bots/api#setgamescore

Параметры

- `user_id` (`int` or `str`) – id пользователя
- `score` (`int`) – Количество очков, должно быть неотрицательным
- `force` (`bool`) – Передайте True, если количество очков могут быть уменьшено. Может быть полезно при исправлении ошибок или бане читеров
- `chat_id` (`int` or `str`) – Обязательный, если не указан inline_message_id. Уникальный id чата или username канала (в формате @channelusername)
- `message_id` (`int`) – Обязательный, если не указан inline_message_id. id отправленного сообщения
- `inline_message_id` (`str`) – Обязательный, если не указаны chat_id и message_id. id inline сообщения
- `disable_edit_message` (`bool`) – Передайте True, если сообщение с игрой должно быть автоматически отредактировано, чтобы отобразить новый результат

Результат
В случае успеха, если сообщение было отправлено ботом, возвращает измененное сообщение (Message), иначе возвращает True.

Тип результата
`types.Message` or `bool`

`async set_message_reaction(`*chat_id: int | str*, *message_id: int*, *reaction: List[*ReactionType*] | None = None*, *is_big: bool | None = None*`) →` bool

Use this method to change the chosen reactions on a message. Service messages can't be reacted to. Automatically forwarded messages from a channel to its discussion group have the same available reactions as messages in the channel. Returns True on success.

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

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username супергруппы или канала (в формате @channelusername)
- `message_id` (`int`) – Identifier of the message to set reaction to
- `reaction` (`list` of *`telebot.types.ReactionType`*) – New list of reaction types to set on the message. Currently, as non-premium users, bots can set up to one reaction per message. A custom emoji reaction can be used if it is either already present on the message or explicitly allowed by chat administrators.
- `is_big` (`bool`) – Pass True to set the reaction with a big animation

Результат
`bool`

`async set_my_commands(commands: List[BotCommand], scope: BotCommandScope | None = None, language_code: str | None = None) → bool`

Используйте этот метод, чтобы изменить список команд бота.

Документация Telegram: https://core.telegram.org/bots/api#setmycommands

Параметры

- `commands` (`list` of *`telebot.types.BotCommand`*) – Список объектов BotCommand. Можно задать не более 100 команд.
- `scope` (*`telebot.types.BotCommandScope`*) – Область видимости команд. По умолчанию BotCommandScopeDefault.
- `language_code` (`str`) – Двухбуквенный языковой код в формате ISO 639-1. Если не задан, изменения коснутся команд для всех пользователей в заданном поле видимости, не имеющих команд на их языке

Результат
True в случае успеха.

Тип результата
`bool`

`async set_my_default_administrator_rights(rights: ChatAdministratorRights = None, for_channels: bool = None) → bool`

Используйте этот метод, чтобы изменить права администратора по умолчанию, запрашиваемые при добавлении бота в группу или канал в качестве администратора. Эти права будут предложены пользователям, но пользователи могут изменить список перед добавлением бота. Возвращает True в случае успеха.

Документация Telegram: https://core.telegram.org/bots/api#setmydefaultadministratorrights

Параметры

- `rights` (*`telebot.types.ChatAdministratorRights`*) – JSON-сериалиованный объект, описывающий новые права администратора по умолчанию. Если не указан, права администратора по умолчанию будут сброшены.
- `for_channels` (`bool`) – Передайте True, чтобы изменить права администратора по умолчанию в каналах. Иначе, будут изменены права администратора по умолчанию для групп и супергрупп.

Результат
True в случае успеха.

Тип результата
`bool`

`async set_my_description(description: str | None = None, language_code: str | None = None)`

Use this method to change the bot's description, which is shown in the chat with the bot if the chat is empty. Returns True on success.

Параметры

- `description` (`str`) – New bot description; 0-512 characters. Pass an empty string to remove the dedicated description for the given language.
- `language_code` (`str`) – A two-letter ISO 639-1 language code. If empty, the description will be applied to all users for whose language there is no dedicated description.

Результат
True в случае успеха.

`async set_my_name(name: str | None = None, language_code: str | None = None)`

Use this method to change the bot's name. Returns True on success.

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

Параметры

- `name` (`str`) – Optional. New bot name; 0-64 characters. Pass an empty string to remove the dedicated name for the given language.
- `language_code` (`str`) – Optional. A two-letter ISO 639-1 language code. If empty, the name will be shown to all users for whose language there is no dedicated name.

Результат
True в случае успеха.

`async set_my_short_description(short_description: str | None = None, language_code: str | None = None)`

Use this method to change the bot's short description, which is shown on the bot's profile page and is sent together with the link when users share the bot. Returns True on success.

Параметры

- `short_description` (`str`) – New short description for the bot; 0-120 characters. Pass an empty string to remove the dedicated short description for the given language.
- `language_code` (`str`) – A two-letter ISO 639-1 language code. If empty, the short description will be applied to all users for whose language there is no dedicated short description.

Результат
True в случае успеха.

`async set_state(user_id: int, state: State | int | str, chat_id: int | None = None)`

Задаёт новое состояние (стейт) пользователя.

Примечание: Вы должны указать и user id и chat id, чтобы задать состояние (стейт) пользователя в чате. Иначе, если вы укажете только user_id, chat_id будет равен user_id, что означает смену состояния (стейта) пользователя в его приватном чате с ботом.

Параметры

- `user_id` (`int`) – id пользователя
- `state` (`int` or `str` or `telebot.types.State`) – новое состояние (стейт). может быть строкой, числом или `telebot.types.State`
- `chat_id` (`int`) – id чата

Результат
None

`async set_sticker_emoji_list(name: str, emoji_list: List[str]) → bool`

Use this method to set the emoji list of a sticker set. Returns True on success.

Параметры

- `name` (`str`) – Имя стикерпака
- `emoji_list` (`list` of `str`) – List of emojis

Результат
Возвращает True в случае успеха.

Тип результата
`bool`

`async set_sticker_keywords(`*sticker: str*, *keywords: List[str] = None*`)` → bool

Use this method to change search keywords assigned to a regular or custom emoji sticker. The sticker must belong to a sticker set created by the bot. Returns True on success.

Параметры

- `sticker` (`str`) – File identifier of the sticker.
- `keywords` (`list` of `str`) – A JSON-serialized list of 0-20 search keywords for the sticker with total length of up to 64 characters

Результат
В случае успеха возвращается True.

Тип результата
`bool`

`async set_sticker_mask_position(`*sticker: str*, *mask_position:* MaskPosition *= None*`)` → bool

Use this method to change the mask position of a mask sticker. The sticker must belong to a sticker set that was created by the bot. Returns True on success.

Параметры

- `sticker` (`str`) – File identifier of the sticker.
- `mask_position` (*`telebot.types.MaskPosition`*) – A JSON-serialized object for position where the mask should be placed on faces.

Результат
Возвращает True в случае успеха.

Тип результата
`bool`

`async set_sticker_position_in_set(`*sticker: str*, *position: int*`)` → bool

Используйте этот метод, чтобы передвинуть стикер в стикерпаке, созданном ботом, на заданную позицию. Возвращает True в случае успеха.

Документация Telegram: https://core.telegram.org/bots/api#setstickerpositioninset

Параметры

- `sticker` (`str`) – id файла стикера
- `position` (`int`) – Новая позиция стикера в стикерпаке, начиная с нуля

Результат
В случае успеха возвращается True.

Тип результата
`bool`

`set_sticker_set_thumb(`***kwargs*`)`

`async set_sticker_set_thumbnail(name: str, user_id: int, thumbnail: Any | str = None, format: str | None = None) → bool`

Используйте этот метод, чтобы задать обложку стикерпака. Анимированные обложки могут быть заданы только для анимированных стикерпаков. Возвращает True в случае успеха.

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

Параметры

- `name` (`str`) – Имя стикерпака
- `user_id` (`int`) – id пользователя
- `thumbnail` (`filelike object`) – A .WEBP or .PNG image with the thumbnail, must be up to 128 kilobytes in size and have a width and height of exactly 100px, or a .TGS animation with a thumbnail up to 32 kilobytes in size (see https://core.telegram.org/stickers#animated-sticker-requirements for animated sticker technical requirements), or a WEBM video with the thumbnail up to 32 kilobytes in size; see https://core.telegram.org/stickers#video-sticker-requirements for video sticker technical requirements. 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. More information on Sending Files ». Animated and video sticker set thumbnails can't be uploaded via HTTP URL. If omitted, then the thumbnail is dropped and the first sticker is used as the thumbnail.

Результат

В случае успеха возвращается True.

Тип результата

`bool`

`async set_sticker_set_title(name: str, title: str) → bool`

Use this method to set the title of a created sticker set. Returns True on success.

Параметры

- `name` (`str`) – Имя стикерпака
- `title` (`str`) – New sticker set title

Результат

Возвращает True в случае успеха.

Тип результата

`bool`

`set_update_listener(func: Awaitable)`

Задаёт функцию-листенер, которая будет вызвана при получении нового апдейта.

Параметры

`func` (`Awaitable`) – Функция-листенер.

Список 27: Пример асинхронного листенера апдейтов.

```
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: str | None = None, certificate: str | Any | None = None, max_connections: int | None = None, allowed_updates: List[str] | None = None, ip_address: str | None = None, drop_pending_updates: bool | None = None, timeout: int | None = None, secret_token: str | None = None*`)` → bool

Используйте этот метод, чтобы задать URL и получать входящие апдейты с помощью вебхука. Как только у бота появляется апдейт, он будет отправлен с помощью HTTPS POST запроса на заданный URL, содержащего JSON-сериализованный Update. В случае неудачного запроса, отправка апдейта будет отменена после разумного числа попыток. Возвращает True в случае успеха.

Если вы хотите удостовериться, что вебхук был задан вами, вы можете задать секретный токен в параметре secret_token. Если указан, запрос с апдейтом будет содержать хедер "X-Telegram-Bot-Api-Secret-Token" с секретным токеном в качестве значения.

Документация Telegram: https://core.telegram.org/bots/api#setwebhook

Параметры

- `url` (`str`, optional) – HTTPS URL для отправки апдейтов. Используйте пустую строку, чтобы удалить вебхук, по умолчанию None
- `certificate` (`str`, optional) – Загрузите публичный ключ вашего SSL сертификата, чтобы корневой сертификат мог быть проверен, по умолчанию None
- `max_connections` (`int`, optional) – Максимально-допустимое количество одновременных HTTPS соединений для доставки апдейтов, 1-100. По умолчанию 40. Используйте меньшие значения для уменьшения нагрузки на ваш сервер и большие значения, чтобы увеличить пропускную способность вашего бота, по умолчанию None
- `allowed_updates` (`list`, optional) – Список видов апдейтов, которые вы хотите получать, в формате JSON. Например, укажите ["message", "edited_channel_post", "callback_query"], чтобы получать апдейты только этих видов. Полный список доступных видов апдейтов - util.update_types. Укажите пустой список, чтобы получать все апдейты, кроме chat_member (по умолчанию). Если не задан, будет использована последняя настройка. Пожалуйста учтите, чтобы этот параметр не влияет на апдейты, отправленные до вызова setWebhooks, поэтому нежелательные апдейты могут быть получены в течение короткого периода времени. По умолчанию None
- `ip_address` (`str`, optional) – Фиксированный IP адрес, который будет использоваться для отправки запросов к вебхукувместо IP адреса, полученного с через DNS, по умолчанию None
- `drop_pending_updates` (`bool`, optional) – Передайте True, чтобы удалить все предшествующие запуску бота апдейты, по умолчанию None
- `timeout` (`int`, optional) – Тайм-аут запроса, по умолчанию None
- `secret_token` (`str`, optional) – Секретный токен для отправки в хедере "X-Telegram-Bot-Api-Secret-Token" в каждом запросе с апдейтом, 1-256 символов. Разрешены только символы A-Z, a-z, 0-9, _ и -. Хедер полезен для, того чтобы удостовериться, что запрос приходитс вебхука, установленного вами. По умолчанию None

Результат
True в случае успеха.

Тип результата
`bool` если запрос был успешным.

`setup_middleware(middleware: BaseMiddleware)`

Настраивает middleware

Примечание: Взгляните на секцию *telebot.asyncio_handler_backends.BaseMiddleware* для получения подробностей.

Параметры
`middleware` (*telebot.asyncio_handler_backends.BaseMiddleware*) – Класс-Middleware.

Результат
None

`shipping_query_handler(func, **kwargs)`

Обрабатывает shipping query. Только для инвойсов с гибкой ценой. В качестве параметра передаёт в декорируемую функцию объект *telebot.types.ShippingQuery*.

Параметры

- `func` (`function`) – Функция, используемая в качестве фильтра
- `kwargs` – Необязательные именованные аргументы(кастомные фильтры)

Результат
None

`async skip_updates()`

Пропускает существующие апдейты. На сервере останется только последний апдейт.

`async stop_message_live_location(chat_id: int | str | None = None, message_id: int | None = None, inline_message_id: str | None = None, reply_markup: InlineKeyboardMarkup | None = None, timeout: int | None = None) → Message`

Используйте этот метод, чтобы остановить обновление live местоположения до истечения live_period. В случае успеха, если сообщение не является inline сообщением,возвращается измененное сообщение (Message), иначе возвращается True.

Документация Telegram: https://core.telegram.org/bots/api#stopmessagelivelocation

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `message_id` (`int`) – Обязательный, если не указан inline_message_id. id сообщения live местоположением, которое нужно остановить
- `inline_message_id` (`str`) – Обязательный, если не указаны chat_id и message_id. id inline сообщения с live местоположением, которое нужно остановить
- `reply_markup` (*telebot.types.InlineKeyboardMarkup* or *telebot.types.ReplyKeyboardMarkup* or *telebot.types.ReplyKeyboardRemove* or *telebot.types.ForceReply*) – JSON-сериализованный объект новой inline клавиатуры.
- `timeout` (`int`) – Таймаут запроса в секундах.

Результат
В случае успеха, если сообщение не является inline сообщением, возвращается измененное сообщение (Message), иначе возвращается True.

Тип результата
telebot.types.Message or bool

`async stop_poll(chat_id: int | str, message_id: int, reply_markup: InlineKeyboardMarkup | None = None) → Poll`

Используйте этот метод, чтобы завершить опрос, отправленный ботом. В случае успеха возвращается завершенный опрос (Poll).

Документация Telegram: https://core.telegram.org/bots/api#stoppoll

Параметры

- `chat_id` (`int` | `str`) – Уникальный id чата или username канала
- `message_id` (`int`) – id сообщения с опросом
- `reply_markup` (`InlineKeyboardMarkup`) – JSON-сериализованный объект новой inline клавиатуры.

Результат
В случае успеха возвращается завершенный опрос (Poll).

Тип результата
`types.Poll`

`async unban_chat_member(chat_id: int | str, user_id: int, only_if_banned: bool | None = False) → bool`

Используйте этот метод, чтобы разбанить ранее кикнутого пользователя в супергруппе или канале. Пользовать не вернется в группу или канал автоматически, но сможет присоединиться с помощью ссылки и т.д. Бот должен быть администратором. По умолчанию, этот метод гарантирует, что после вызова, пользователь не является участником чата, но может присоединиться. Поэтому если пользовать является участником чата, он будет кикнут, но не забанен. Если вы хотите изменить это поведение, используйте параметр only_if_banned.

Документация Telegram: https://core.telegram.org/bots/api#unbanchatmember

Параметры

- `chat_id` (`int` or `str`) – Уникальный id группы или username супергруппы или канала (в формате @username)
- `user_id` (`int`) – Уникальный id сделавшего запрос пользователя
- `only_if_banned` (`bool`) – Ничего не делать, если пользователь не забанен

Результат
True в случае успеха

Тип результата
`bool`

`async unban_chat_sender_chat(chat_id: int | str, sender_chat_id: int | str) → bool`

Используйте этот метод, чтобы разбанить ране забаненный канал в супергруппе или канала. Бот должен быть администратором и иметь соответствующие права администратора. Возвращает True в случае успеха.

Документация Telegram: https://core.telegram.org/bots/api#unbanchatsenderchat

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `sender_chat_id` (`int` or `str`) – Уникальный id чата.

Результат
True в случае успеха.

Тип результата
`bool`

`async unhide_general_forum_topic(chat_id: int | str) → bool`

Используйте этот метод, чтобы сделать топик „General“ видимым в супергруппе с топиками. Бот должен быть администратором чата и иметь права администратора can_manage_topics. Возвращает True в случае успеха.

Документация Telegram: https://core.telegram.org/bots/api#unhidegeneralforumtopic

Параметры
`chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)

`async unpin_all_chat_messages(chat_id: int | str) → bool`

Используйте этот метод, что открепить все закрепленные сообщения в супергруппе. Бот должен быть администратором чата и иметь соответствующие права администратора. Возвращает True в случае успеха.

Документация Telegram: https://core.telegram.org/bots/api#unpinallchatmessages

Параметры
`chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)

Результат
True в случае успеха.

Тип результата
`bool`

`async unpin_all_forum_topic_messages(chat_id: str | int, message_thread_id: int) → bool`

Используйте этот метод, что открепить все закрепленные сообщения в топике. Бот должен быть администратором чата и иметь права администратора can_pin_messages в супергруппе. Возвращает True в случае успеха.

Документация Telegram: https://core.telegram.org/bots/api#unpinallforumtopicmessages

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `message_thread_id` (`int`) – id топика

Результат
В случае успеха возвращается True.

Тип результата
`bool`

`async unpin_all_general_forum_topic_messages(chat_id: int | str) → bool`

Use this method to clear the list of pinned messages in a General 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#unpinAllGeneralForumTopicMessages

Параметры
`chat_id` (`int` | `str`) – Unique identifier for the target chat or username of chat

Результат
В случае успеха возвращается True.

Тип результата
`bool`

`async unpin_chat_message(chat_id: int | str, message_id: int | None = None)` → bool

Используйте этот метод, что открепить закрепленное сообщение в супергруппе. Бот должен быть администратором чата и иметь соответствующие права администратора. Возвращает True в случае успеха.

Документация Telegram: https://core.telegram.org/bots/api#unpinchatmessage

Параметры

- `chat_id` (`int` or `str`) – Уникальный id чата или username канала (в формате @channelusername)
- `message_id` (`int`) – Int: id сообщения, которое нужно открепить

Результат
True в случае успеха.

Тип результата
`bool`

`async upload_sticker_file(user_id: int, png_sticker: Any | str = None, sticker: InputFile | None = None, sticker_format: str | None = None)` → *File*

Используйте этот метод, чтобы загрузить .png стикер, чтобы позже использовать в методах createNewStickerSet и addStickerToSet (может быть использован несколько раз). Возвращает загруженный файл (File) в случае успеха.

Документация Telegram: https://core.telegram.org/bots/api#uploadstickerfile

Параметры

- `user_id` (`int`) – id пользователя, создавшего стикерпак
- `png_sticker` (`filelike object`) – DEPRECATED: 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.
- `sticker` (*`telebot.types.InputFile`*) – A file with the sticker in .WEBP, .PNG, .TGS, or .WEBM format. See https://core.telegram.org/stickers for technical requirements. More information on Sending Files »
- `sticker_format` (`str`) – One of «static», «animated», «video».

Результат
В случае успеха возвращается отправленный файл.

Тип результата
`telebot.types.File`

`property user`

class telebot.async_telebot.ExceptionHandler

Базовые классы: object

Класс для обработки исключений во время поллинга

async handle(*exception*)

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

Базовые классы: object

Класс для (next step|reply) хендлеров

Файл asyncio_filters

class telebot.asyncio_filters.AdvancedCustomFilter

Базовые классы: ABC

Базовый класс Advanced Custom Filter. Создайте класс наследник с методом check(). Принимает два параметра, возвращает bool: True - фильтр пройден, False - фильтр не пройден. message: класс Message text: значение фильтра, полученное в хендлере

Классы наследники должны иметь статический атрибут (property) .key

Список 28: Пример создания 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*)

Выполнить проверку.

key: str = None

class telebot.asyncio_filters.ChatFilter

Базовые классы: *AdvancedCustomFilter*

Проверяет, является ли chat_id заданным.

Список 29: Пример использования этого фильтра:

```
@bot.message_handler(chat_id=[99999])
# your function
```

key: str = 'chat_id'

class telebot.asyncio_filters.ForwardFilter

Базовые классы: *SimpleCustomFilter*

Проверяет, является ли сообщение пересланным из канала или группы.

Список 30: Пример использования этого фильтра:

```
@bot.message_handler(is_forwarded=True)
# your function
```

key: str = 'is_forwarded'

class telebot.asyncio_filters.IsAdminFilter(*bot*)

Базовые классы: *SimpleCustomFilter*

Проверяет, является ли пользователь администратором / владельцем чата.

Список 31: Пример использования этого фильтра:

```
@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*

Фильтр для проверки, состоит ли строка только из цифр.

Список 32: Пример использования этого фильтра:

```
@bot.message_handler(is_digit=True)
# your function
```

key: str = 'is_digit'

class telebot.asyncio_filters.IsReplyFilter

Базовые классы: *SimpleCustomFilter*

Проверяет, является ли сообщение ответом (reply).

Список 33: Пример использования этого фильтра:

```
@bot.message_handler(is_reply=True)
# your function
```

key: str = 'is_reply'

class telebot.asyncio_filters.LanguageFilter

Базовые классы: *AdvancedCustomFilter*

Проверяет language_code пользователя.

Список 34: Пример использования этого фильтра:

```
@bot.message_handler(language_code=['ru'])
# your function
```

key: str = 'language_code'

class telebot.asyncio_filters.SimpleCustomFilter

Базовые классы: ABC

Базовый класс Simple Custom Filter. Создайте класс наследник с методом check(). Принимает только сообщение, возвращает bool, который сравнивается с заданным в хендлере.

Классы наследники должны иметь статический атрибут (property) .key

Список 35: Пример создания simple custom filter.

```
class ForwardFilter(SimpleCustomFilter):
    # Check whether message was forwarded from channel or group.
    key = 'is_forwarded'

    def check(self, message):
        return message.forward_date is not None
```

async check(*message*)

Выполнить проверку.

key: str = None

class telebot.asyncio_filters.StateFilter(*bot*)

Базовые классы: *AdvancedCustomFilter*

Фильтр для проверки состояния (стейта).

Список 36: Пример использования этого фильтра:

```
@bot.message_handler(state=1)
# your function
```

key: str = 'state'

class telebot.asyncio_filters.TextContainsFilter

Базовые классы: *AdvancedCustomFilter*

Фильтр для проверки текста сообщения. key: text

Список 37: Пример использования этого фильтра:

```
# 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: str | None = None, contains: list | tuple | None = None, starts_with: str | list | tuple | None = None, ends_with: str | list | tuple | None = None, ignore_case: bool = False*)

Базовые классы: object

Advanced текстовый фильтр для проверки (types.Message, types.CallbackQuery, types.InlineQuery, types.Poll)

пример использования в examples/asynchronous_telebot/custom_filters/advanced_text_filter.py

Параметры

- `equals` (`str`) – строка, True если текст объекта идентичен заданной строке
- `contains` (*`list[str] or tuple[str]`*) – list[str] или tuple[str], True если хотя бы один из элементов есть в тексте
- `starts_with` (`str`) – string, True если текст объекта начинается с заданной строки
- `ends_with` (`str`) – string, True если текст объекта начинается с заданной строки
- `ignore_case` (`bool`) – bool (по умолчанию False), независимый от регистра

Исключение
`ValueError` – если было задано некорректное значение параметра

Результат
None

`class telebot.asyncio_filters.TextMatchFilter`

Базовые классы: *`AdvancedCustomFilter`*

Фильтр для проверки текста сообщения.

Список 38: Пример использования этого фильтра:

```
@bot.message_handler(text=['account'])
# your function
```

`key: str = 'text'`

`class telebot.asyncio_filters.TextStartsFilter`

Базовые классы: *`AdvancedCustomFilter`*

Фильтр для проверки, начинается ли сообщение с заданного текста.

Список 39: Пример использования этого фильтра:

```
# Will work if message.text starts with 'sir'.
@bot.message_handler(text_startswith='sir')
# your function
```

`key: str = 'text_startswith'`

Файл asyncio_handler_backends

Файл со всеми классами middleware и states.

`class telebot.asyncio_handler_backends.BaseMiddleware`

Базовые классы: `object`

Базовый класс для middleware. Ваши middleware должны быть унаследованы от этого класса.

Задайте update_sensitive=True если хотите получать разные апдейты в разных функциях. Например, если вы хотите обрабатывать pre_process для апдейтов вида message, вам нужно будет создать функцию pre_process_message и т.д. Аналогично для post_process.

Список 40: Пример класса middleware

```
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):
        # 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`

Класс для отмены апдейтов. Просто верните экземпляр этого класса в middleware, чтобы пропустить апдейт. Апдейт пропустит хендлер и исполнение post_process в middleware.

`class telebot.asyncio_handler_backends.ContinueHandling`

Базовые классы: `object`

Класс для продолжения обработки апдейта в хендлерах. Просто верните экземпляр этого класса в хендлерах, чтобы продолжить обработку.

Список 41: Пример использования 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`

Класс для пропуска хендлеров. Просто верните экземпляр этого класса в middleware, чтобы пропустить хендлер. Апдейт попадёт в post_process, но пропустит исполнение хендлера.

class telebot.asyncio_handler_backends.State

Базовые классы: object

Класс, представляющий состояние (стейт).

```
class MyStates(StatesGroup):
    my_state = State() # returns my_state:State string.
```

class telebot.asyncio_handler_backends.StatesGroup

Базовые классы: object

Класс, представляющий похожие состояния (стейты).

```
class MyStates(StatesGroup):
    my_state = State() # returns my_state:State string.
```

classmethod state_list()

Расширения

1.3.6 Фабрика callback data

Файл callback_data

Файл фабрики callback data.

class telebot.callback_data.CallbackData(*parts, prefix: str, sep=':')

Базовые классы: object

Фабрика Callback data. Этот класс поможет вам в работе с CallbackQuery

filter(**config) → CallbackDataFilter

Сгенерировать фильтр

Параметры

config – заданные именованные параметры будут проверены в CallbackQuery.data

Результат

Класс CallbackDataFilter

new(*args, **kwargs) → str

Сгенерировать callback data

Параметры

- args – позиционные параметры экземпляра CallbackData
- kwargs – именованные параметры

Результат

str

parse(callback_data: str) → Dict[str, str]

Получить данные из callback data

Параметры

callback_data – string, примените к telebot.types.CallbackQuery, чтобы преобразовать callback_data из строки (str) в словарь (dict)

Результат
словарь (dict), полученный из callback data

`class telebot.callback_data.CallbackDataFilter(factory, config: Dict[str, str])`

Базовые классы: `object`

Фильтр для CallbackData.

`check(query)` → bool

Проверяет, соответствует ли query.data заданной конфигурации

Параметры
`query` (`telebot.types.CallbackQuery`) – telebot.types.CallbackQuery

Результат
True, если query.data соответствует заданной конфигурации

Тип результата
bool

1.3.7 Утилиты

Файл util

`telebot.util.antiflood(function: Callable, *args, number_retries=5, **kwargs)`

Используйте эту функцию в циклах, чтобы избежать ошибки TooManyRequests. Пример:

```
from telebot.util import antiflood
for chat_id in chat_id_list:
msg = antiflood(bot.send_message, chat_id, text)
```

Параметры

- `function` (*:obj:int*) – Вызываемая функция
- `number_retries` – Number of retries to send
- `args` (`tuple`) – Аргументы, для передачи в функцию
- `kwargs` (`dict`) – Именованные аргументы для передачи в функцию

Результат
None

`telebot.util.chunks(lst, n)`

Генерирует последовательные части списка, состоящие из n элементов.

`telebot.util.content_type_media = ['text', 'animation', 'audio', 'document', 'photo', 'sticker', 'story', 'video', 'video_note', 'voice', 'contact', 'dice', 'game', 'poll', 'venue', 'location', 'invoice', 'successful_payment', 'connected_website', 'passport_data', 'web_app_data']`

Содержит все виды медиа.

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', 'message_auto_delete_timer_changed', 'migrate_to_chat_id', 'migrate_from_chat_id', 'pinned_message', 'users_shared', 'chat_shared', 'write_access_allowed', 'proximity_alert_triggered', 'forum_topic_created', 'forum_topic_edited', 'forum_topic_closed', 'forum_topic_reopened', 'general_forum_topic_hidden', 'general_forum_topic_unhidden', 'giveaway_created', 'giveaway', 'giveaway_winners', 'giveaway_completed', 'video_chat_scheduled', 'video_chat_started', 'video_chat_ended', 'video_chat_participants_invited']

Содержит все виды сервисных сообщений, такие как *User joined the group.*

telebot.util.escape(*text: str*) → str | None

Заменяет следующие символы в *text* ("&" на „&", „<" на „<" и „>" на „>").

Параметры
text – Текст для замены символов

Результат
Отформатированный текст

telebot.util.extract_arguments(*text: str*) → str

Возвращает аргументы команды.

Список 42: Примеры:

```
extract_arguments("/get name"): 'name'
extract_arguments("/get"): ''
extract_arguments("/get@botName name"): 'name'
```

Параметры
text (str) – Строка для извлечения аргументов команды

Результат
Аргументы, если *text* является командой (согласно is_command), в остальных случаях None.

Тип результата
str или None

telebot.util.extract_command(*text: str*) → str | None

Извлекает команду из *text* (исключает „/") если *text* является командой (см. is_command). Если *text* не является командой, эта функция возвращает None.

Список 43: Примеры:

```
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) – Строка, из которой нужно извлечь команду

Результат
Команда, если *text* является командой (согласно is_command), в остальных случаях

None.

Тип результата
`str` или `None`

`telebot.util.generate_random_token()` → str

Генерирует рандомный токен, состоящий из латинских букв и цифр длиной 16 символов.

Результат
Сгенерированный токен

Тип результата
`str`

`telebot.util.is_bytes(var)` → bool

Возвращает True если полученный объект является bytes.

Параметры
`var` (`object`) – Объект для проверки

Результат
True, если полученный объект является bytes.

Тип результата
`bool`

`telebot.util.is_command(text: str)` → bool

Проверяет, является ли *text* командой. Команды в Telegram начинаются с символа „/“.

Параметры
`text` (`str`) – Текст для проверки.

Результат
True, если *text* является командой, иначе False.

Тип результата
`bool`

`telebot.util.is_dict(var)` → bool

Возвращает True, если полученный объект является словарём (dict).

Параметры
`var` (`object`) – Объект для проверки

Результат
True, если полученный объект является словарём (dict).

Тип результата
`bool`

`telebot.util.is_pil_image(var)` → bool

Возвращает True, если полученный объект является PIL.Image.Image.

Параметры
`var` (`object`) – Объект для проверки

Результат
True, если полученный объект является PIL.Image.Image.

Тип результата
`bool`

`telebot.util.is_string(var)` → bool

Возвращает True, если полученный объект является строкой (str).

`telebot.util.parse_web_app_data(token: str, raw_init_data: str)`

Обрабатывает данные, полученные от web app.

Параметры

- `token` (`str`) – Токен бота
- `raw_init_data` (`str`) – Необработанные данные

Результат

Обработанные данные

`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*

Возвращает reply markup из словаря следующего формата: {„text“: kwargs}. Удобно использовать вместо постоянного использования „btn1 = InlineKeyboardButton(...)“ „btn2 = InlineKeyboardButton(...)“

Пример:

Список 44: Используя quick_markup:

```
from telebot.util import quick_markup

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`) – Словарь, содержащий все кнопки для создания reply markup в следующем формате: {text: kwargs} {str:}
- `row_width` (`int`) – number of *`telebot.types.InlineKeyboardButton`* objects on each row

Результат
InlineKeyboardMarkup

Тип результата
`types.InlineKeyboardMarkup`

`telebot.util.smart_split(`*text: str*, *chars_per_string: int = 4096*`)` → List[str]

Разбивает строку на несколько, каждая из которых будет не длиннее *characters_per_string*. Удобно использовать для разбиения одного гигантского сообщения на несколько. Если *chars_per_string* > 4096: *chars_per_string* = 4096. Разбивает строку по „n“, „. „ или „ „ именно в таком порядке.

Параметры

- `text` (`str`) – Текст для разбиения
- `chars_per_string` (`int`) – Максимальное количество символов в части текста, на которые он будет разбит.

Результат
Список частей разбитого текста.

Тип результата
`list` of `str`

`telebot.util.split_string(`*text: str*, *chars_per_string: int*`)` → List[str]

Разбивает одну строку на несколько, каждая из которых будет не длиннее *characters_per_string*. Удобно использовать для разбиения одного гигантского сообщения на несколько.

Параметры

- `text` (`str`) – Текст для разбиения
- `chars_per_string` (`int`) – Количество символов в одной строке, на которые будет разбит текст.

Результат
Список частей разбитого текста.

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', 'message_reaction', 'message_reaction_count',
'chat_boost', 'removed_chat_boost', 'business_connection', 'business_message',
'edited_business_message', 'deleted_business_messages']
```

Все виды апдейтов, рекомендуется использовать в качестве параметра allowed_updates функции polling.

`telebot.util.user_link(`*user:* User, *include_id: bool = False*`)` → str

Возвращает HTML ссылку на пользователя. Удобно использовать для отчетов. Важно: Не забудьте установить значение „HTML“ в parse_mode!

Список 45: Пример:

```
bot.send_message(your_user_id, user_link(message.from_user) + ' started the bot!',␣
↪parse_mode='HTML')
```

Примечание: Вы можете использовать formatting.* во всех остальных вариантах форматирования(bold, italic, links, и прочее). Этот метод сохранён для обратной совместимости, рекомендуется использовать formatting.* для большего количества вариантов.

Параметры

- `user` (*telebot.types.User*) – Пользователь (не id пользователя)
- `include_id` (`bool`) – Добавить id пользователя

Результат
Ссылка на пользователя в формате HTML

Тип результата
`str`

`telebot.util.validate_web_app_data(token: str, raw_init_data: str)`

Проверяет данные, полученные от web app.

Параметры

- `token` (`str`) – Токен бота
- `raw_init_data` (`str`) – Необработанные данные

Результат
Обработанные данные

`telebot.util.webhook_google_functions(bot, request)`

Endpoint вебхука для Google Cloud Functions FaaS.

Параметры

- `bot` (*telebot.TeleBot* or *telebot.async_telebot.AsyncTeleBot*) – Инстанс бота
- `request` (`flask.Request`) – HTTP-запрос

Результат
Объект, полученный в качестве ответа

1.3.8 Параметры форматирования

Функции форматирования Markdown & HTML.

Added in version 4.5.1.

`telebot.formatting.apply_html_entities(text: str, entities: List | None, custom_subs: Dict[str, str] | None) → str`

Author: @sviat9440 Updaters: @badiboy, @EgorKhabarov Message: «*Test* parse _formatting_, [url](https://example.com), [text_mention](tg://user?id=123456) and mention @username»

Список 46: Example:

```
apply_html_entities(text, entities)
>> "<b>Test</b> parse <i>formatting</i>, <a href="https://example.com">url</a>, <a␣
↪href="tg://user?id=123456">text_mention</a> and mention @username"
```

Custom subs:
You can customize the substitutes. By default, there is no substitute for the entities: hashtag, bot_command, email. You can add or modify substitute an existing entity.

Список 47: Example:

```
apply_html_entities(
    text,
    entities,
    {"bold": "<strong class="example">{text}</strong>", "italic": "<i class="example
↪">{text}</i>", "mention": "<a href={url}>{text}</a>"},
)
>> "<strong class="example">Test</strong> parse <i class="example">formatting</i>,
↪<a href="https://example.com">url</a> and <a href="tg://user?id=123456">text_
↪mention</a> and mention <a href="https://t.me/username">@username</a>"
```

`telebot.formatting.escape_html`(*content: str*) → str

Пропускает HTML символы в HTML строке.

Параметры
`content` (`str`) – HTML строка, которую нужно пропустить.

Результат
Пропускаемая строка.

Тип результата
`str`

`telebot.formatting.escape_markdown`(*content: str*) → str

Пропускает Markdown символы в Markdown строке.

Credits to: simonsmh

Параметры
`content` (`str`) – Markdown строка, которую нужно пропустить.

Результат
Пропускаемая строка.

Тип результата
`str`

`telebot.formatting.format_text`(**args*, *separator='\n'*)

Преобразовывает набор строк в одну.

```
format_text( # just an example
    mbold('Hello'),
    mitalic('World')
)
```

Параметры

- `args` (`str`) – Строки для преобразования.
- `separator` (`str`) – Символ для разделения строк.

Результат
Преобразованная строка.

Тип результата
`str`

`telebot.formatting.hbold(`*content: str*, *escape: bool | None = True*`)` → str

Возвращает выделенную жирным шрифтом HTML строку.

Параметры

- `content` (`str`) – Строка для выделения жирным шрифтом.
- `escape` (`bool`) – True если вам нужно пропустить спец. символы. По умолчанию True.

Результат
Преобразованная строка.

Тип результата
`str`

`telebot.formatting.hcite(`*content: str*, *escape: bool | None = True*`)` → str

Returns a html-formatted block-quotation string.

Параметры

- `content` (`str`) – Строка для выделения жирным шрифтом.
- `escape` (`bool`) – True если вам нужно пропустить спец. символы. По умолчанию True.

Результат
Преобразованная строка.

Тип результата
`str`

`telebot.formatting.hcode(`*content: str*, *escape: bool | None = True*`)` → str

Возвращает выделенную как код HTML строку.

Параметры

- `content` (`str`) – Строка для выделения как код.
- `escape` (`bool`) – True если вам нужно пропустить спец. символы. По умолчанию True.

Результат
Преобразованная строка.

Тип результата
`str`

`telebot.formatting.hide_link(`*url: str*`)` → str

Делает невидимым URL изображения.

Параметры
`url` (`str`) – URL изображения.

Результат
Невидимый URL.

Тип результата
`str`

`telebot.formatting.hitalic(`*content: str*, *escape: bool | None = True*`)` → str

Возвращает выделенную курсивом HTML строку.

Параметры

- `content` (`str`) – Строка для выделения курсивом.
- `escape` (`bool`) – True если вам нужно пропустить спец. символы. По умолчанию True.

Результат
Преобразованная строка.

Тип результата
`str`

`telebot.formatting.hlink(`*content: str*, *url: str*, *escape: bool | None = True*`)` → str

Возвращает HTML строку с гиперссылкой.

Параметры

- `content` (`str`) – Строка для добавления гиперссылки.
- `url` (`str`) – URL для создания гиперссылки.
- `escape` (`bool`) – True если вам нужно пропустить спец. символы. По умолчанию True.

Результат
Преобразованная строка.

Тип результата
`str`

`telebot.formatting.hpre(`*content: str*, *escape: bool | None = True*, *language: str = ''*`)` → str

Возвращает предварительно отформатированную HTML строку.

Параметры

- `content` (`str`) – Строка для предварительного форматирования.
- `escape` (`bool`) – True если вам нужно пропустить спец. символы. По умолчанию True.

Результат
Преобразованная строка.

Тип результата
`str`

`telebot.formatting.hspoiler(`*content: str*, *escape: bool | None = True*`)` → str

Возвращает выделенную как спойлер HTML строку.

Параметры

- `content` (`str`) – Строка для выделения как спойлер.
- `escape` (`bool`) – True если вам нужно пропустить спец. символы. По умолчанию True.

Результат
Преобразованная строка.

Тип результата
`str`

`telebot.formatting.hstrikethrough(`*content: str*, *escape: bool | None = True*`)` → str

Возвращает зачеркнутую HTML строку.

Параметры

- `content` (`str`) – Строка для зачеркивания.
- `escape` (`bool`) – True если вам нужно пропустить спец. символы. По умолчанию True.

Результат
Преобразованная строка.

Тип результата
`str`

`telebot.formatting.hunderline(`*content: str*, *escape: bool | None = True*`)` → str

Возвращает подчеркнутую HTML строку.

Параметры

- `content` (`str`) – Строка для подчёркивания.
- `escape` (`bool`) – True если вам нужно пропустить спец. символы. По умолчанию True.

Результат
Преобразованная строка.

Тип результата
`str`

`telebot.formatting.mbold(`*content: str*, *escape: bool | None = True*`)` → str

Возвращает выделенную жирным шрифтом Markdown строку.

Параметры

- `content` (`str`) – Строка для выделения жирным шрифтом.
- `escape` (`bool`) – True если вам нужно пропустить спец. символы. По умолчанию True.

Результат
Преобразованная строка.

Тип результата
`str`

`telebot.formatting.mcite(`*content: str*, *escape: bool | None = True*`)` → str

Returns a Markdown-formatted block-quotation string.

Параметры

- `content` (`str`) – Строка для выделения жирным шрифтом.
- `escape` (`bool`) – True если вам нужно пропустить спец. символы. По умолчанию True.

Результат
Преобразованная строка.

Тип результата
`str`

`telebot.formatting.mcode(`*content: str*, *language: str = ''*, *escape: bool | None = True*`)` → str

Возвращает выделенную как код Markdown строку.

Параметры

- `content` (`str`) – Строка для выделения как код.
- `escape` (`bool`) – True если вам нужно пропустить спец. символы. По умолчанию True.

Результат
Преобразованная строка.

Тип результата
`str`

`telebot.formatting.mitalic(`*content: str*, *escape: bool | None = True*`)` → str

Возвращает выделенную курсивом Markdown строку.

Параметры

- `content` (`str`) – Строка для выделения курсивом.
- `escape` (`bool`) – True если вам нужно пропустить спец. символы. По умолчанию True.

Результат
Преобразованная строка.

Тип результата
`str`

`telebot.formatting.mlink(`*content: str*, *url: str*, *escape: bool | None = True*`)` → str

Возвращает Markdown строку с гиперссылкой.

Параметры

- `content` (`str`) – Строка для добавления гиперссылки.
- `url` (*`str`*) – URL для создания гиперссылки.
- `escape` (`bool`) – True если вам нужно пропустить спец. символы. По умолчанию True.

Результат
Преобразованная строка.

Тип результата
`str`

`telebot.formatting.mspoiler(`*content: str*, *escape: bool | None = True*`)` → str

Возвращает выделенную как спойлер Markdown строку.

Параметры

- `content` (`str`) – Строка для выделения как спойлер.
- `escape` (`bool`) – True если вам нужно пропустить спец. символы. По умолчанию True.

Результат
Преобразованная строка.

Тип результата
`str`

`telebot.formatting.mstrikethrough(`*content: str*, *escape: bool | None = True*`)` → str

Возвращает зачеркнутую Markdown строку.

Параметры

- `content` (`str`) – Строка для зачеркивания.
- `escape` (`bool`) – True если вам нужно пропустить спец. символы. По умолчанию True.

Результат
Преобразованная строка.

Тип результата
`str`

`telebot.formatting.munderline(`*content: str*, *escape: bool | None = True*`)` → str

Возвращает подчеркнутую Markdown строку.

Параметры

- `content` (`str`) – Строка для подчёркивания.
- `escape` (`bool`) – True если вам нужно пропустить спец. символы. По умолчанию True.

Результат
Преобразованная строка.

Тип результата
`str`

Глава 2

Ссылки

- genindex
- modindex
- search

Содержание модулей Python

t

telebot, 103
telebot.async_telebot, 194
telebot.asyncio_filters, 278
telebot.asyncio_handler_backends, 281
telebot.callback_data, 283
telebot.custom_filters, 189
telebot.formatting, 289
telebot.handler_backends, 193
telebot.types, 4
telebot.util, 284

Алфавитный указатель

A

add() (*метод telebot.types.InlineKeyboardMarkup*), 39
add() (*метод telebot.types.Poll*), 83
add() (*метод telebot.types.ReplyKeyboardMarkup*), 87
add_custom_filter() (*метод telebot.async_telebot.AsyncTeleBot*), 195
add_custom_filter() (*метод telebot.TeleBot*), 104
add_data() (*метод telebot.async_telebot.AsyncTeleBot*), 196
add_data() (*метод telebot.TeleBot*), 105
add_price() (*метод telebot.types.ShippingOption*), 90
add_sticker_to_set() (*метод telebot.async_telebot.AsyncTeleBot*), 196
add_sticker_to_set() (*метод telebot.TeleBot*), 105
AdvancedCustomFilter (*класс в telebot.asyncio_filters*), 278
AdvancedCustomFilter (*класс в telebot.custom_filters*), 189
Animation (*класс в telebot.types*), 4
answer_callback_query() (*метод telebot.async_telebot.AsyncTeleBot*), 197
answer_callback_query() (*метод telebot.TeleBot*), 105
answer_inline_query() (*метод telebot.async_telebot.AsyncTeleBot*), 197
answer_inline_query() (*метод telebot.TeleBot*), 106
answer_pre_checkout_query() (*метод telebot.async_telebot.AsyncTeleBot*), 198
answer_pre_checkout_query() (*метод telebot.TeleBot*), 107
answer_shipping_query() (*метод telebot.async_telebot.AsyncTeleBot*), 199
answer_shipping_query() (*метод telebot.TeleBot*), 107
answer_web_app_query() (*метод telebot.async_telebot.AsyncTeleBot*), 199
answer_web_app_query() (*метод telebot.TeleBot*), 108
antiflood() (*в модуле telebot.util*), 284
apply_html_entities() (*в модуле telebot.formatting*), 289
approve_chat_join_request() (*метод telebot.async_telebot.AsyncTeleBot*), 199
approve_chat_join_request() (*метод telebot.TeleBot*), 108
AsyncTeleBot (*класс в telebot.async_telebot*), 194
Audio (*класс в telebot.types*), 4

B

ban_chat_member() (*метод telebot.async_telebot.AsyncTeleBot*), 200
ban_chat_member() (*метод telebot.TeleBot*), 108
ban_chat_sender_chat() (*метод telebot.async_telebot.AsyncTeleBot*), 200
ban_chat_sender_chat() (*метод telebot.TeleBot*), 109
BaseMiddleware (*класс в telebot.asyncio_handler_backends*), 281
BaseMiddleware (*класс в telebot.handler_backends*), 193
Birthdate (*класс в telebot.types*), 5
BotCommand (*класс в telebot.types*), 5
BotCommandScope (*класс в telebot.types*), 5

BotCommandScopeAllChatAdministrators (*класс в telebot.types*), 6
BotCommandScopeAllGroupChats (*класс в telebot.types*), 7
BotCommandScopeAllPrivateChats (*класс в telebot.types*), 7
BotCommandScopeChat (*класс в telebot.types*), 7
BotCommandScopeChatAdministrators (*класс в telebot.types*), 8
BotCommandScopeChatMember (*класс в telebot.types*), 8
BotCommandScopeDefault (*класс в telebot.types*), 8
BotDescription (*класс в telebot.types*), 8
BotName (*класс в telebot.types*), 9
BotShortDescription (*класс в telebot.types*), 9
business_connection_handler() (*метод telebot.async_ telebot.AsyncTeleBot*), 201
business_connection_handler() (*метод telebot.TeleBot*), 109
business_message_handler() (*метод telebot.async_ telebot.AsyncTeleBot*), 201
business_message_handler() (*метод telebot.TeleBot*), 110
BusinessConnection (*класс в telebot.types*), 9
BusinessIntro (*класс в telebot.types*), 10
BusinessLocation (*класс в telebot.types*), 10
BusinessMessagesDeleted (*класс в telebot.types*), 10
BusinessOpeningHours (*класс в telebot.types*), 11
BusinessOpeningHoursInterval (*класс в telebot.types*), 11

C

callback_query_handler() (*метод telebot.async_ telebot.AsyncTeleBot*), 202
callback_query_handler() (*метод telebot.TeleBot*), 110
CallbackData (*класс в telebot.callback_ data*), 283
CallbackDataFilter (*класс в telebot.callback_ data*), 284
CallbackQuery (*класс в telebot.types*), 11
can_manage_voice_chats (*telebot.types.ChatMember property*), 20
CancelUpdate (*класс в telebot.asyncio_ handler_ backends*), 282
CancelUpdate (*класс в telebot.handler_ backends*), 193
channel_post_handler() (*метод telebot.async_ telebot.AsyncTeleBot*), 202
channel_post_handler() (*метод telebot.TeleBot*), 111
Chat (*класс в telebot.types*), 12
chat_boost_handler() (*метод telebot.async_ telebot.AsyncTeleBot*), 202
chat_boost_handler() (*метод telebot.TeleBot*), 111
chat_join_request_handler() (*метод telebot.async_ telebot.AsyncTeleBot*), 202
chat_join_request_handler() (*метод telebot.TeleBot*), 111
chat_member_handler() (*метод telebot.async_ telebot.AsyncTeleBot*), 203
chat_member_handler() (*метод telebot.TeleBot*), 111
ChatAdministratorRights (*класс в telebot.types*), 15
ChatBoost (*класс в telebot.types*), 16
ChatBoostAdded (*класс в telebot.types*), 16
ChatBoostRemoved (*класс в telebot.types*), 17
ChatBoostSource (*класс в telebot.types*), 17
ChatBoostSourceGiftCode (*класс в telebot.types*), 17
ChatBoostSourceGiveaway (*класс в telebot.types*), 17
ChatBoostSourcePremium (*класс в telebot.types*), 18
ChatBoostUpdated (*класс в telebot.types*), 18
ChatFilter (*класс в telebot.asyncio_ filters*), 278
ChatFilter (*класс в telebot.custom_ filters*), 189
ChatInviteLink (*класс в telebot.types*), 18
ChatJoinRequest (*класс в telebot.types*), 19
ChatLocation (*класс в telebot.types*), 20
ChatMember (*класс в telebot.types*), 20
ChatMemberAdministrator (*класс в telebot.types*), 20
ChatMemberBanned (*класс в telebot.types*), 22
ChatMemberLeft (*класс в telebot.types*), 23
ChatMemberMember (*класс в telebot.types*), 23
ChatMemberOwner (*класс в telebot.types*), 24
ChatMemberRestricted (*класс в telebot.types*), 24
ChatMemberUpdated (*класс в telebot.types*), 26
ChatPermissions (*класс в telebot.types*), 26
ChatPhoto (*класс в telebot.types*), 27
ChatShared (*класс в telebot.types*), 28
check() (*метод telebot.asyncio_ filters.AdvancedCustomFilter*), 278
check() (*метод telebot.asyncio_ filters.SimpleCustomFilter*), 280
check() (*метод telebot.callback_ data.CallbackDataFilter*), 284
check() (*метод telebot.custom_ filters.AdvancedCustomFilter*),

189
check() (*метод telebot.custom_filters.SimpleCustomFilter*), 191
chosen_inline_handler() (*метод telebot.async_telebot.AsyncTeleBot*), 203
chosen_inline_handler() (*метод telebot.TeleBot*), 112
ChosenInlineResult (*класс в telebot.types*), 28
chunks() (*в модуле telebot.util*), 284
clear_reply_handlers() (*метод telebot.TeleBot*), 112
clear_reply_handlers_by_message_id() (*метод telebot.TeleBot*), 112
clear_step_handler() (*метод telebot.TeleBot*), 112
clear_step_handler_by_chat_id() (*метод telebot.TeleBot*), 112
close() (*метод telebot.async_telebot.AsyncTeleBot*), 203
close() (*метод telebot.TeleBot*), 112
close_forum_topic() (*метод telebot.async_telebot.AsyncTeleBot*), 203
close_forum_topic() (*метод telebot.TeleBot*), 113
close_general_forum_topic() (*метод telebot.async_telebot.AsyncTeleBot*), 204
close_general_forum_topic() (*метод telebot.TeleBot*), 113
close_session() (*метод telebot.async_telebot.AsyncTeleBot*), 204
Contact (*класс в telebot.types*), 29
contains_masks (*telebot.types.StickerSet property*), 92
content_type_media (*в модуле telebot.util*), 284
content_type_service (*в модуле telebot.util*), 284
ContinueHandling (*класс в telebot.asyncio_handler_backends*), 282
ContinueHandling (*класс в telebot.handler_backends*), 193
convert_input_sticker() (*метод telebot.types.InputSticker*), 65
copy_message() (*метод telebot.async_telebot.AsyncTeleBot*), 204
copy_message() (*метод telebot.TeleBot*), 113
copy_messages() (*метод telebot.async_telebot.AsyncTeleBot*), 205
copy_messages() (*метод telebot.TeleBot*), 114
create_chat_invite_link() (*метод telebot.async_telebot.AsyncTeleBot*), 206
create_chat_invite_link() (*метод telebot.TeleBot*), 115
create_forum_topic() (*метод telebot.async_telebot.AsyncTeleBot*), 206
create_forum_topic() (*метод telebot.TeleBot*), 115
create_invoice_link() (*метод telebot.async_telebot.AsyncTeleBot*), 206
create_invoice_link() (*метод telebot.TeleBot*), 116
create_new_sticker_set() (*метод telebot.async_telebot.AsyncTeleBot*), 208
create_new_sticker_set() (*метод telebot.TeleBot*), 117

D

decline_chat_join_request() (*метод telebot.async_telebot.AsyncTeleBot*), 209
decline_chat_join_request() (*метод telebot.TeleBot*), 118
delete_chat_photo() (*метод telebot.async_telebot.AsyncTeleBot*), 209
delete_chat_photo() (*метод telebot.TeleBot*), 118
delete_chat_sticker_set() (*метод telebot.async_telebot.AsyncTeleBot*), 210
delete_chat_sticker_set() (*метод telebot.TeleBot*), 119
delete_forum_topic() (*метод telebot.async_telebot.AsyncTeleBot*), 210
delete_forum_topic() (*метод telebot.TeleBot*), 119
delete_message() (*метод telebot.async_telebot.AsyncTeleBot*), 210
delete_message() (*метод telebot.TeleBot*), 119
delete_messages() (*метод telebot.async_telebot.AsyncTeleBot*), 211
delete_messages() (*метод telebot.TeleBot*), 120
delete_my_commands() (*метод telebot.async_telebot.AsyncTeleBot*), 211
delete_my_commands() (*метод telebot.TeleBot*), 120
delete_state() (*метод telebot.async_telebot.AsyncTeleBot*),

211
delete_state() (*метод telebot.TeleBot*), 120
delete_sticker_from_set() (*метод telebot.async_ telebot.AsyncTeleBot*), 211
delete_sticker_from_set() (*метод telebot.TeleBot*), 121
delete_sticker_set() (*метод telebot.async_ telebot.AsyncTeleBot*), 212
delete_sticker_set() (*метод telebot.TeleBot*), 121
delete_webhook() (*метод telebot.async_ telebot.AsyncTeleBot*), 212
delete_webhook() (*метод telebot.TeleBot*), 121
deleted_business_messages_handler() (*метод telebot.async_ telebot.AsyncTeleBot*), 212
deleted_business_messages_handler() (*метод telebot.TeleBot*), 121
Dice (*класс в telebot.types*), 29
Dictionaryable (*класс в telebot.types*), 30
difference (*telebot.types.ChatMemberUpdated property*), 26
disable_save_next_step_handlers() (*метод telebot.TeleBot*), 121
disable_save_reply_handlers() (*метод telebot.TeleBot*), 122
Document (*класс в telebot.types*), 30
download_file() (*метод telebot.async_ telebot.AsyncTeleBot*), 212
download_file() (*метод telebot.TeleBot*), 122

E

edit_chat_invite_link() (*метод telebot.async_ telebot.AsyncTeleBot*), 213
edit_chat_invite_link() (*метод telebot.TeleBot*), 122
edit_forum_topic() (*метод telebot.async_ telebot.AsyncTeleBot*), 213
edit_forum_topic() (*метод telebot.TeleBot*), 122
edit_general_forum_topic() (*метод telebot.async_ telebot.AsyncTeleBot*), 214
edit_general_forum_topic() (*метод telebot.TeleBot*), 123
edit_message_caption() (*метод telebot.async_ telebot.AsyncTeleBot*), 214
edit_message_caption() (*метод telebot.TeleBot*), 123
edit_message_live_location() (*метод telebot.async_ telebot.AsyncTeleBot*), 214
edit_message_live_location() (*метод telebot.TeleBot*), 124
edit_message_media() (*метод telebot.async_ telebot.AsyncTeleBot*), 215
edit_message_media() (*метод telebot.TeleBot*), 125
edit_message_reply_markup() (*метод telebot.async_ telebot.AsyncTeleBot*), 216
edit_message_reply_markup() (*метод telebot.TeleBot*), 125
edit_message_text() (*метод telebot.async_ telebot.AsyncTeleBot*), 216
edit_message_text() (*метод telebot.TeleBot*), 126
edited_business_message_handler() (*метод telebot.async_ telebot.AsyncTeleBot*), 217
edited_business_message_handler() (*метод telebot.TeleBot*), 126
edited_channel_post_handler() (*метод telebot.async_ telebot.AsyncTeleBot*), 217
edited_channel_post_handler() (*метод telebot.TeleBot*), 127
edited_message_handler() (*метод telebot.async_ telebot.AsyncTeleBot*), 218
edited_message_handler() (*метод telebot.TeleBot*), 127
enable_save_next_step_handlers() (*метод telebot.TeleBot*), 127
enable_save_reply_handlers() (*метод telebot.TeleBot*), 128
enable_saving_states() (*метод telebot.async_ telebot.AsyncTeleBot*), 218
enable_saving_states() (*метод telebot.TeleBot*), 128
escape() (*в модуле telebot.util*), 285
escape_html() (*в модуле telebot.formatting*), 290
escape_markdown() (*в модуле telebot.formatting*), 290
ExceptionHandler (*класс в telebot*), 103
ExceptionHandler (*класс в telebot.async_ telebot*), 277
export_chat_invite_link() (*метод telebot.async_ telebot.AsyncTeleBot*), 218
export_chat_invite_link() (*метод telebot.TeleBot*), 128

ExternalReplyInfo (*класс в telebot.types*), 30
extract_arguments() (*в модуле telebot.util*), 285
extract_command() (*в модуле telebot.util*), 285

F

file (*telebot.types.InputFile property*), 58
File (*класс в telebot.types*), 32
filter() (*метод telebot.callback_data.CallbackData*), 283
ForceReply (*класс в telebot.types*), 32
format_text() (*в модуле telebot.formatting*), 290
ForumTopic (*класс в telebot.types*), 33
ForumTopicClosed (*класс в telebot.types*), 33
ForumTopicCreated (*класс в telebot.types*), 33
ForumTopicEdited (*класс в telebot.types*), 34
ForumTopicReopened (*класс в telebot.types*), 34
forward_date (*telebot.types.Message property*), 77
forward_from (*telebot.types.Message property*), 77
forward_from_chat (*telebot.types.Message property*), 77
forward_from_message_id (*telebot.types.Message property*), 77
forward_message() (*метод telebot.async_telebot.AsyncTeleBot*), 218
forward_message() (*метод telebot.TeleBot*), 128
forward_messages() (*метод telebot.async_telebot.AsyncTeleBot*), 219
forward_messages() (*метод telebot.TeleBot*), 129
forward_sender_name (*telebot.types.Message property*), 77
forward_signature (*telebot.types.Message property*), 77
ForwardFilter (*класс в telebot.asyncio_filters*), 278
ForwardFilter (*класс в telebot.custom_filters*), 190
full_name (*telebot.types.User property*), 96

G

Game (*класс в telebot.types*), 34
GameHighScore (*класс в telebot.types*), 35
GeneralForumTopicHidden (*класс в telebot.types*), 35
GeneralForumTopicUnhidden (*класс в telebot.types*), 35
generate_random_token() (*в модуле telebot.util*), 286
get_business_connection() (*метод telebot.async_telebot.AsyncTeleBot*), 220
get_business_connection() (*метод telebot.TeleBot*), 129
get_chat() (*метод telebot.async_telebot.AsyncTeleBot*), 220
get_chat() (*метод telebot.TeleBot*), 130
get_chat_administrators() (*метод telebot.async_telebot.AsyncTeleBot*), 220
get_chat_administrators() (*метод telebot.TeleBot*), 130
get_chat_member() (*метод telebot.async_telebot.AsyncTeleBot*), 220
get_chat_member() (*метод telebot.TeleBot*), 130
get_chat_member_count() (*метод telebot.async_telebot.AsyncTeleBot*), 221
get_chat_member_count() (*метод telebot.TeleBot*), 130
get_chat_members_count() (*метод telebot.async_telebot.AsyncTeleBot*), 221
get_chat_members_count() (*метод telebot.TeleBot*), 131
get_chat_menu_button() (*метод telebot.async_telebot.AsyncTeleBot*), 221
get_chat_menu_button() (*метод telebot.TeleBot*), 131
get_custom_emoji_stickers() (*метод telebot.async_telebot.AsyncTeleBot*), 221
get_custom_emoji_stickers() (*метод telebot.TeleBot*), 131
get_file() (*метод telebot.async_telebot.AsyncTeleBot*), 221
get_file() (*метод telebot.TeleBot*), 131
get_file_url() (*метод telebot.async_telebot.AsyncTeleBot*), 222
get_file_url() (*метод telebot.TeleBot*), 131
get_forum_topic_icon_stickers() (*метод telebot.async_telebot.AsyncTeleBot*), 222
get_forum_topic_icon_stickers() (*метод telebot.TeleBot*), 132
get_game_high_scores() (*метод telebot.async_telebot.AsyncTeleBot*), 222
get_game_high_scores() (*метод telebot.TeleBot*), 132
get_me() (*метод telebot.async_telebot.AsyncTeleBot*), 223
get_me() (*метод telebot.TeleBot*), 132
get_my_commands() (*метод telebot.async_telebot.AsyncTeleBot*), 223
get_my_commands() (*метод telebot.TeleBot*), 132
get_my_default_administrator_rights() (*метод*

telebot.async_ telebot.AsyncTeleBot), 223
get_my_default_administrator_rights() (метод telebot.TeleBot), 133
get_my_description() (метод telebot.async_ telebot.AsyncTeleBot), 223
get_my_description() (метод telebot.TeleBot), 133
get_my_name() (метод telebot.async_ telebot.AsyncTeleBot), 223
get_my_name() (метод telebot.TeleBot), 133
get_my_short_description() (метод telebot.async_ telebot.AsyncTeleBot), 224
get_my_short_description() (метод telebot.TeleBot), 133
get_state() (метод telebot.async_ telebot.AsyncTeleBot), 224
get_state() (метод telebot.TeleBot), 134
get_sticker_set() (метод telebot.async_ telebot.AsyncTeleBot), 224
get_sticker_set() (метод telebot.TeleBot), 134
get_updates() (метод telebot.async_ telebot.AsyncTeleBot), 224
get_updates() (метод telebot.TeleBot), 134
get_user_chat_boosts() (метод telebot.async_ telebot.AsyncTeleBot), 225
get_user_chat_boosts() (метод telebot.TeleBot), 135
get_user_profile_photos() (метод telebot.async_ telebot.AsyncTeleBot), 225
get_user_profile_photos() (метод telebot.TeleBot), 135
get_webhook_info() (метод telebot.async_ telebot.AsyncTeleBot), 225
get_webhook_info() (метод telebot.TeleBot), 135
Giveaway (класс в telebot.types), 35
GiveawayCompleted (класс в telebot.types), 36
GiveawayCreated (класс в telebot.types), 36
GiveawayWinners (класс в telebot.types), 36

H

handle() (метод telebot.async_ telebot.ExceptionHandler), 278
handle() (метод telebot.ExceptionHandler), 103
Handler (класс в telebot), 103
Handler (класс в telebot.async_ telebot), 278
hbold() (в модуле telebot.formatting), 291
hcite() (в модуле telebot.formatting), 291
hcode() (в модуле telebot.formatting), 291
hide_general_forum_topic() (метод telebot.async_ telebot.AsyncTeleBot), 226
hide_general_forum_topic() (метод telebot.TeleBot), 136
hide_link() (в модуле telebot.formatting), 291
hitalic() (в модуле telebot.formatting), 292
hlink() (в модуле telebot.formatting), 292
hpre() (в модуле telebot.formatting), 292
hspoiler() (в модуле telebot.formatting), 292
hstrikethrough() (в модуле telebot.formatting), 293
html_caption (telebot.types.Message property), 77
html_text (telebot.types.Message property), 77
html_text (telebot.types.TextQuote property), 94
hunderline() (в модуле telebot.formatting), 293

I

InaccessibleMessage (класс в telebot.types), 37
infinity_polling() (метод telebot.async_ telebot.AsyncTeleBot), 226
infinity_polling() (метод telebot.TeleBot), 136
inline_handler() (метод telebot.async_ telebot.AsyncTeleBot), 227
inline_handler() (метод telebot.TeleBot), 136
InlineKeyboardButton (класс в telebot.types), 38
InlineKeyboardMarkup (класс в telebot.types), 39
InlineQuery (класс в telebot.types), 40
InlineQueryResultArticle (класс в telebot.types), 40
InlineQueryResultAudio (класс в telebot.types), 41
InlineQueryResultBase (класс в telebot.types), 42
InlineQueryResultCachedAudio (класс в telebot.types), 43
InlineQueryResultCachedBase (класс в telebot.types), 43
InlineQueryResultCachedDocument (класс в telebot.types), 43
InlineQueryResultCachedGif (класс в telebot.types), 44
InlineQueryResultCachedMpeg4Gif (класс в telebot.types), 45
InlineQueryResultCachedPhoto (класс в telebot.types), 46
InlineQueryResultCachedSticker (класс в telebot.types), 46
InlineQueryResultCachedVideo (класс в telebot.types), 47

InlineQueryResultCachedVoice (*класс в telebot.types*), 47
InlineQueryResultContact (*класс в telebot.types*), 48
InlineQueryResultDocument (*класс в telebot.types*), 49
InlineQueryResultGame (*класс в telebot.types*), 50
InlineQueryResultGif (*класс в telebot.types*), 50
InlineQueryResultLocation (*класс в telebot.types*), 51
InlineQueryResultMpeg4Gif (*класс в telebot.types*), 52
InlineQueryResultPhoto (*класс в telebot.types*), 53
InlineQueryResultsButton (*класс в telebot.types*), 57
InlineQueryResultVenue (*класс в telebot.types*), 54
InlineQueryResultVideo (*класс в telebot.types*), 55
InlineQueryResultVoice (*класс в telebot.types*), 56
InputContactMessageContent (*класс в telebot.types*), 57
InputFile (*класс в telebot.types*), 58
InputInvoiceMessageContent (*класс в telebot.types*), 58
InputLocationMessageContent (*класс в telebot.types*), 60
InputMedia (*класс в telebot.types*), 60
InputMediaAnimation (*класс в telebot.types*), 60
InputMediaAudio (*класс в telebot.types*), 61
InputMediaDocument (*класс в telebot.types*), 62
InputMediaPhoto (*класс в telebot.types*), 63
InputMediaVideo (*класс в telebot.types*), 63
InputSticker (*класс в telebot.types*), 64
InputTextMessageContent (*класс в telebot.types*), 65
InputVenueMessageContent (*класс в telebot.types*), 65
Invoice (*класс в telebot.types*), 66
is_animated (*telebot.types.StickerSet property*), 92
is_bytes() (*в модуле telebot.util*), 286
is_command() (*в модуле telebot.util*), 286
is_dict() (*в модуле telebot.util*), 286
is_pil_image() (*в модуле telebot.util*), 286
is_string() (*в модуле telebot.util*), 286
is_video (*telebot.types.StickerSet property*), 92
IsAdminFilter (*класс в telebot.asyncio_filters*), 279
IsAdminFilter (*класс в telebot.custom_filters*), 190
IsDigitFilter (*класс в telebot.asyncio_filters*), 279
IsDigitFilter (*класс в telebot.custom_filters*), 190
IsReplyFilter (*класс в telebot.asyncio_filters*), 279
IsReplyFilter (*класс в telebot.custom_filters*), 190

J

JsonDeserializable (*класс в telebot.types*), 66
JsonSerializable (*класс в telebot.types*), 66

K

key (*атрибут telebot.asyncio_filters.AdvancedCustomFilter*), 278
key (*атрибут telebot.asyncio_filters.ChatFilter*), 278
key (*атрибут telebot.asyncio_filters.ForwardFilter*), 279
key (*атрибут telebot.asyncio_filters.IsAdminFilter*), 279
key (*атрибут telebot.asyncio_filters.IsDigitFilter*), 279
key (*атрибут telebot.asyncio_filters.IsReplyFilter*), 279
key (*атрибут telebot.asyncio_filters.LanguageFilter*), 279
key (*атрибут telebot.asyncio_filters.SimpleCustomFilter*), 280
key (*атрибут telebot.asyncio_filters.StateFilter*), 280
key (*атрибут telebot.asyncio_filters.TextContainsFilter*), 280
key (*атрибут telebot.asyncio_filters.TextMatchFilter*), 281
key (*атрибут telebot.asyncio_filters.TextStartsFilter*), 281
key (*атрибут telebot.custom_filters.AdvancedCustomFilter*), 189
key (*атрибут telebot.custom_filters.ChatFilter*), 190
key (*атрибут telebot.custom_filters.ForwardFilter*), 190
key (*атрибут telebot.custom_filters.IsAdminFilter*), 190
key (*атрибут telebot.custom_filters.IsDigitFilter*), 190
key (*атрибут telebot.custom_filters.IsReplyFilter*), 190
key (*атрибут telebot.custom_filters.LanguageFilter*), 191
key (*атрибут telebot.custom_filters.SimpleCustomFilter*), 191
key (*атрибут telebot.custom_filters.StateFilter*), 191
key (*атрибут telebot.custom_filters.TextContainsFilter*), 191
key (*атрибут telebot.custom_filters.TextMatchFilter*), 192
key (*атрибут telebot.custom_filters.TextStartsFilter*), 192
KeyboardButton (*класс в telebot.types*), 66
KeyboardButtonPollType (*класс в telebot.types*), 67
KeyboardButtonRequestChat (*класс в telebot.types*), 68
KeyboardButtonRequestUser (*класс в*

telebot.types), 69
KeyboardButtonRequestUsers (класс в telebot.types), 69
kick_chat_member() (метод telebot.async_telebot.AsyncTeleBot), 227
kick_chat_member() (метод telebot.TeleBot), 137

L

LabeledPrice (класс в telebot.types), 69
LanguageFilter (класс в telebot.asyncio_filters), 279
LanguageFilter (класс в telebot.custom_filters), 191
leave_chat() (метод telebot.async_telebot.AsyncTeleBot), 227
leave_chat() (метод telebot.TeleBot), 137
LinkPreviewOptions (класс в telebot.types), 70
load_next_step_handlers() (метод telebot.TeleBot), 137
load_reply_handlers() (метод telebot.TeleBot), 137
Location (класс в telebot.types), 70
log_out() (метод telebot.async_telebot.AsyncTeleBot), 227
log_out() (метод telebot.TeleBot), 137
LoginUrl (класс в telebot.types), 71

M

MaskPosition (класс в telebot.types), 71
max_row_keys (атрибут telebot.types.InlineKeyboardMarkup), 40
max_row_keys (атрибут telebot.types.ReplyKeyboardMarkup), 87
mbold() (в модуле telebot.formatting), 293
mcite() (в модуле telebot.formatting), 293
mcode() (в модуле telebot.formatting), 294
MenuButton (класс в telebot.types), 72
MenuButtonCommands (класс в telebot.types), 72
MenuButtonDefault (класс в telebot.types), 72
MenuButtonWebApp (класс в telebot.types), 72
Message (класс в telebot.types), 73
message_handler() (метод telebot.async_telebot.AsyncTeleBot), 227
message_handler() (метод telebot.TeleBot), 138
message_reaction_count_handler() (метод telebot.async_telebot.AsyncTeleBot), 228
message_reaction_count_handler() (метод telebot.TeleBot), 139
message_reaction_handler() (метод telebot.async_telebot.AsyncTeleBot), 229
message_reaction_handler() (метод telebot.TeleBot), 139
MessageAutoDeleteTimerChanged (класс в telebot.types), 78
MessageEntity (класс в telebot.types), 78
MessageID (класс в telebot.types), 79
MessageOrigin (класс в telebot.types), 79
MessageOriginChannel (класс в telebot.types), 80
MessageOriginChat (класс в telebot.types), 80
MessageOriginHiddenUser (класс в telebot.types), 80
MessageOriginUser (класс в telebot.types), 80
MessageReactionCountUpdated (класс в telebot.types), 80
MessageReactionUpdated (класс в telebot.types), 81
middleware_handler() (метод telebot.TeleBot), 139
mitalic() (в модуле telebot.formatting), 294
mlink() (в модуле telebot.formatting), 294
module
  telebot, 103
  telebot.async_telebot, 194
  telebot.asyncio_filters, 278
  telebot.asyncio_handler_backends, 281
  telebot.callback_data, 283
  telebot.custom_filters, 189
  telebot.formatting, 289
  telebot.handler_backends, 193
  telebot.types, 4
  telebot.util, 284
mspoiler() (в модуле telebot.formatting), 294
mstrikethrough() (в модуле telebot.formatting), 295
munderline() (в модуле telebot.formatting), 295
my_chat_member_handler() (метод telebot.async_telebot.AsyncTeleBot), 229
my_chat_member_handler() (метод telebot.TeleBot), 140

N

new() (метод telebot.callback_data.CallbackData), 283
new_chat_member (telebot.types.Message property), 77

O

OrderInfo (класс в telebot.types), 81

P

parse() (метод telebot.callback_data.CallbackData), 283

parse_chat() (*метод класса telebot.types.Message*), 78
parse_entities() (*метод класса telebot.types.Game*), 35
parse_entities() (*метод класса telebot.types.Message*), 78
parse_photo() (*метод класса telebot.types.Game*), 35
parse_photo() (*метод класса telebot.types.Message*), 78
parse_web_app_data() (*в модуле telebot.util*), 287
PhotoSize (*класс в telebot.types*), 82
pil_image_to_file() (*в модуле telebot.util*), 287
pin_chat_message() (*метод telebot.async_telebot.AsyncTeleBot*), 229
pin_chat_message() (*метод telebot.TeleBot*), 140
Poll (*класс в telebot.types*), 82
poll_answer_handler() (*метод telebot.async_telebot.AsyncTeleBot*), 229
poll_answer_handler() (*метод telebot.TeleBot*), 140
poll_handler() (*метод telebot.async_telebot.AsyncTeleBot*), 230
poll_handler() (*метод telebot.TeleBot*), 140
PollAnswer (*класс в telebot.types*), 83
polling() (*метод telebot.async_telebot.AsyncTeleBot*), 230
polling() (*метод telebot.TeleBot*), 141
PollOption (*класс в telebot.types*), 83
post_process() (*метод telebot.asyncio_handler_backends.BaseMiddleware*), 282
post_process() (*метод telebot.handler_backends.BaseMiddleware*), 193
pre_checkout_query_handler() (*метод telebot.async_telebot.AsyncTeleBot*), 231
pre_checkout_query_handler() (*метод telebot.TeleBot*), 141
pre_process() (*метод telebot.asyncio_handler_backends.BaseMiddleware*), 282
pre_process() (*метод telebot.handler_backends.BaseMiddleware*), 193
PreCheckoutQuery (*класс в telebot.types*), 84
process_new_updates() (*метод telebot.async_telebot.AsyncTeleBot*), 231
process_new_updates() (*метод telebot.TeleBot*), 142
promote_chat_member() (*метод telebot.async_telebot.AsyncTeleBot*), 231
promote_chat_member() (*метод telebot.TeleBot*), 142
ProximityAlertTriggered (*класс в telebot.types*), 84

Q

quick_markup() (*в модуле telebot.util*), 287

R

ReactionCount (*класс в telebot.types*), 84
ReactionType (*класс в telebot.types*), 85
ReactionTypeCustomEmoji (*класс в telebot.types*), 85
ReactionTypeEmoji (*класс в telebot.types*), 85
register_business_connection_handler() (*метод telebot.async_telebot.AsyncTeleBot*), 232
register_business_connection_handler() (*метод telebot.TeleBot*), 143
register_business_message_handler() (*метод telebot.async_telebot.AsyncTeleBot*), 233
register_business_message_handler() (*метод telebot.TeleBot*), 143
register_callback_query_handler() (*метод telebot.async_telebot.AsyncTeleBot*), 233
register_callback_query_handler() (*метод telebot.TeleBot*), 144
register_channel_post_handler() (*метод telebot.async_telebot.AsyncTeleBot*), 233
register_channel_post_handler() (*метод telebot.TeleBot*), 144
register_chat_boost_handler() (*метод telebot.async_telebot.AsyncTeleBot*), 234
register_chat_boost_handler() (*метод telebot.TeleBot*), 144
register_chat_join_request_handler() (*метод telebot.async_telebot.AsyncTeleBot*), 234
register_chat_join_request_handler() (*метод telebot.TeleBot*), 145
register_chat_member_handler() (*метод telebot.async_telebot.AsyncTeleBot*), 234
register_chat_member_handler() (*метод telebot.TeleBot*), 145
register_chosen_inline_handler() (*метод telebot.async_telebot.AsyncTeleBot*), 234
register_chosen_inline_handler() (*метод telebot.TeleBot*), 145

register_deleted_business_messages_handler() (*метод telebot.async_ telebot.AsyncTeleBot*), 235
register_deleted_business_messages_handler() (*метод telebot.TeleBot*), 146
register_edited_business_message_handler() (*метод telebot.async_ telebot.AsyncTeleBot*), 235
register_edited_business_message_handler() (*метод telebot.TeleBot*), 146
register_edited_channel_post_handler() (*метод telebot.async_ telebot.AsyncTeleBot*), 235
register_edited_channel_post_handler() (*метод telebot.TeleBot*), 146
register_edited_message_handler() (*метод telebot.async_ telebot.AsyncTeleBot*), 236
register_edited_message_handler() (*метод telebot.TeleBot*), 147
register_for_reply() (*метод telebot.TeleBot*), 147
register_for_reply_by_message_id() (*метод telebot.TeleBot*), 147
register_inline_handler() (*метод telebot.async_ telebot.AsyncTeleBot*), 236
register_inline_handler() (*метод telebot.TeleBot*), 148
register_message_handler() (*метод telebot.async_ telebot.AsyncTeleBot*), 237
register_message_handler() (*метод telebot.TeleBot*), 148
register_message_reaction_count_handler() (*метод telebot.async_ telebot.AsyncTeleBot*), 237
register_message_reaction_count_handler() (*метод telebot.TeleBot*), 149
register_message_reaction_handler() (*метод telebot.async_ telebot.AsyncTeleBot*), 237
register_message_reaction_handler() (*метод telebot.TeleBot*), 149
register_middleware_handler() (*метод telebot.TeleBot*), 149
register_my_chat_member_handler() (*метод telebot.async_ telebot.AsyncTeleBot*), 238
register_my_chat_member_handler() (*метод telebot.TeleBot*), 149
register_next_step_handler() (*метод telebot.TeleBot*), 150
register_next_step_handler_by_chat_id() (*метод telebot.TeleBot*), 150
register_poll_answer_handler() (*метод telebot.async_ telebot.AsyncTeleBot*), 238
register_poll_answer_handler() (*метод telebot.TeleBot*), 150
register_poll_handler() (*метод telebot.async_ telebot.AsyncTeleBot*), 238
register_poll_handler() (*метод telebot.TeleBot*), 151
register_pre_checkout_query_handler() (*метод telebot.async_ telebot.AsyncTeleBot*), 238
register_pre_checkout_query_handler() (*метод telebot.TeleBot*), 151
register_removed_chat_boost_handler() (*метод telebot.async_ telebot.AsyncTeleBot*), 239
register_removed_chat_boost_handler() (*метод telebot.TeleBot*), 151
register_shipping_query_handler() (*метод telebot.async_ telebot.AsyncTeleBot*), 239
register_shipping_query_handler() (*метод telebot.TeleBot*), 151
remove_webhook() (*метод telebot.async_ telebot.AsyncTeleBot*), 239
remove_webhook() (*метод telebot.TeleBot*), 152
removed_chat_boost_handler() (*метод telebot.async_ telebot.AsyncTeleBot*), 239
removed_chat_boost_handler() (*метод telebot.TeleBot*), 152
reopen_forum_topic() (*метод telebot.async_ telebot.AsyncTeleBot*), 240
reopen_forum_topic() (*метод telebot.TeleBot*), 152
reopen_general_forum_topic() (*метод telebot.async_ telebot.AsyncTeleBot*), 240
reopen_general_forum_topic() (*метод telebot.TeleBot*), 152
replace_sticker_in_set() (*метод telebot.async_ telebot.AsyncTeleBot*), 240
replace_sticker_in_set() (*метод telebot.TeleBot*), 153
REPLY_MARKUP_TYPES (*в модуле telebot*), 103
reply_to() (*метод telebot.async_ telebot.AsyncTeleBot*), 240
reply_to() (*метод telebot.TeleBot*), 153
ReplyKeyboardMarkup (*класс в telebot.types*), 85
ReplyKeyboardRemove (*класс в telebot.types*), 87
ReplyParameters (*класс в telebot.types*), 88
reset_data() (*метод telebot.async_ telebot.AsyncTeleBot*), 241

reset_data() (*метод telebot.TeleBot*), 153
restrict_chat_member() (*метод telebot.async_telebot.AsyncTeleBot*), 241
restrict_chat_member() (*метод telebot.TeleBot*), 153
retrieve_data() (*метод telebot.async_telebot.AsyncTeleBot*), 242
retrieve_data() (*метод telebot.TeleBot*), 154
revoke_chat_invite_link() (*метод telebot.async_telebot.AsyncTeleBot*), 242
revoke_chat_invite_link() (*метод telebot.TeleBot*), 155
row() (*метод telebot.types.InlineKeyboardMarkup*), 40
row() (*метод telebot.types.ReplyKeyboardMarkup*), 87
run_webhooks() (*метод telebot.async_telebot.AsyncTeleBot*), 242
run_webhooks() (*метод telebot.TeleBot*), 155

S

send_animation() (*метод telebot.async_telebot.AsyncTeleBot*), 243
send_animation() (*метод telebot.TeleBot*), 156
send_audio() (*метод telebot.async_telebot.AsyncTeleBot*), 245
send_audio() (*метод telebot.TeleBot*), 157
send_chat_action() (*метод telebot.async_telebot.AsyncTeleBot*), 246
send_chat_action() (*метод telebot.TeleBot*), 159
send_contact() (*метод telebot.async_telebot.AsyncTeleBot*), 247
send_contact() (*метод telebot.TeleBot*), 159
send_dice() (*метод telebot.async_telebot.AsyncTeleBot*), 248
send_dice() (*метод telebot.TeleBot*), 160
send_document() (*метод telebot.async_telebot.AsyncTeleBot*), 249
send_document() (*метод telebot.TeleBot*), 161
send_game() (*метод telebot.async_telebot.AsyncTeleBot*), 250
send_game() (*метод telebot.TeleBot*), 163
send_invoice() (*метод telebot.async_telebot.AsyncTeleBot*), 251
send_invoice() (*метод telebot.TeleBot*), 163
send_location() (*метод telebot.async_telebot.AsyncTeleBot*), 253
send_location() (*метод telebot.TeleBot*), 165
send_media_group() (*метод telebot.async_telebot.AsyncTeleBot*), 254
send_media_group() (*метод telebot.TeleBot*), 166
send_message() (*метод telebot.async_telebot.AsyncTeleBot*), 255
send_message() (*метод telebot.TeleBot*), 167
send_photo() (*метод telebot.async_telebot.AsyncTeleBot*), 256
send_photo() (*метод telebot.TeleBot*), 168
send_poll() (*метод telebot.async_telebot.AsyncTeleBot*), 257
send_poll() (*метод telebot.TeleBot*), 169
send_sticker() (*метод telebot.async_telebot.AsyncTeleBot*), 259
send_sticker() (*метод telebot.TeleBot*), 171
send_venue() (*метод telebot.async_telebot.AsyncTeleBot*), 260
send_venue() (*метод telebot.TeleBot*), 172
send_video() (*метод telebot.async_telebot.AsyncTeleBot*), 261
send_video() (*метод telebot.TeleBot*), 173
send_video_note() (*метод telebot.async_telebot.AsyncTeleBot*), 262
send_video_note() (*метод telebot.TeleBot*), 174
send_voice() (*метод telebot.async_telebot.AsyncTeleBot*), 264
send_voice() (*метод telebot.TeleBot*), 175
SentWebAppMessage (*класс в telebot.types*), 88
set_chat_administrator_custom_title() (*метод telebot.async_telebot.AsyncTeleBot*), 265
set_chat_administrator_custom_title() (*метод telebot.TeleBot*), 176
set_chat_description() (*метод telebot.async_telebot.AsyncTeleBot*), 265
set_chat_description() (*метод telebot.TeleBot*), 177

set_chat_menu_button() (*метод telebot.async_ telebot.AsyncTeleBot*), 265
set_chat_menu_button() (*метод telebot.TeleBot*), 177
set_chat_permissions() (*метод telebot.async_ telebot.AsyncTeleBot*), 266
set_chat_permissions() (*метод telebot.TeleBot*), 177
set_chat_photo() (*метод telebot.async_ telebot.AsyncTeleBot*), 266
set_chat_photo() (*метод telebot.TeleBot*), 178
set_chat_sticker_set() (*метод telebot.async_ telebot.AsyncTeleBot*), 267
set_chat_sticker_set() (*метод telebot.TeleBot*), 178
set_chat_title() (*метод telebot.async_ telebot.AsyncTeleBot*), 267
set_chat_title() (*метод telebot.TeleBot*), 178
set_custom_emoji_sticker_set_thumbnail() (*метод telebot.async_ telebot.AsyncTeleBot*), 267
set_custom_emoji_sticker_set_thumbnail() (*метод telebot.TeleBot*), 179
set_game_score() (*метод telebot.async_ telebot.AsyncTeleBot*), 268
set_game_score() (*метод telebot.TeleBot*), 179
set_message_reaction() (*метод telebot.async_ telebot.AsyncTeleBot*), 268
set_message_reaction() (*метод telebot.TeleBot*), 180
set_my_commands() (*метод telebot.async_ telebot.AsyncTeleBot*), 268
set_my_commands() (*метод telebot.TeleBot*), 180
set_my_default_administrator_rights() (*метод telebot.async_ telebot.AsyncTeleBot*), 269
set_my_default_administrator_rights() (*метод telebot.TeleBot*), 180
set_my_description() (*метод telebot.async_ telebot.AsyncTeleBot*), 269
set_my_description() (*метод telebot.TeleBot*), 181
set_my_name() (*метод telebot.async_ telebot.AsyncTeleBot*), 270
set_my_name() (*метод telebot.TeleBot*), 181
set_my_short_description() (*метод telebot.async_ telebot.AsyncTeleBot*), 270
set_my_short_description() (*метод telebot.TeleBot*), 181
set_state() (*метод telebot.async_ telebot.AsyncTeleBot*), 270
set_state() (*метод telebot.TeleBot*), 182
set_sticker_emoji_list() (*метод telebot.async_ telebot.AsyncTeleBot*), 270
set_sticker_emoji_list() (*метод telebot.TeleBot*), 182
set_sticker_keywords() (*метод telebot.async_ telebot.AsyncTeleBot*), 271
set_sticker_keywords() (*метод telebot.TeleBot*), 182
set_sticker_mask_position() (*метод telebot.async_ telebot.AsyncTeleBot*), 271
set_sticker_mask_position() (*метод telebot.TeleBot*), 182
set_sticker_position_in_set() (*метод telebot.async_ telebot.AsyncTeleBot*), 271
set_sticker_position_in_set() (*метод telebot.TeleBot*), 183
set_sticker_set_thumb() (*метод telebot.async_ telebot.AsyncTeleBot*), 271
set_sticker_set_thumb() (*метод telebot.TeleBot*), 183
set_sticker_set_thumbnail() (*метод telebot.async_ telebot.AsyncTeleBot*), 271
set_sticker_set_thumbnail() (*метод telebot.TeleBot*), 183
set_sticker_set_title() (*метод telebot.async_ telebot.AsyncTeleBot*), 272
set_sticker_set_title() (*метод telebot.TeleBot*), 184
set_update_listener() (*метод telebot.async_ telebot.AsyncTeleBot*), 272
set_update_listener() (*метод telebot.TeleBot*), 184
set_webhook() (*метод telebot.async_ telebot.AsyncTeleBot*), 273
set_webhook() (*метод telebot.TeleBot*), 184
setup_middleware() (*метод*

telebot.async_ telebot.AsyncTeleBot), 274
setup_middleware() (метод telebot.TeleBot), 185
SharedUser (класс в telebot.types), 89
shipping_query_handler() (метод telebot.async_ telebot.AsyncTeleBot), 274
shipping_query_handler() (метод telebot.TeleBot), 185
ShippingAddress (класс в telebot.types), 89
ShippingOption (класс в telebot.types), 90
ShippingQuery (класс в telebot.types), 90
SimpleCustomFilter (класс в telebot.asyncio_ filters), 279
SimpleCustomFilter (класс в telebot.custom_ filters), 191
skip_updates() (метод telebot.async_ telebot.AsyncTeleBot), 274
SkipHandler (класс в telebot.asyncio_ handler_ backends), 282
SkipHandler (класс в telebot.handler_ backends), 194
smart_split() (в модуле telebot.util), 288
split_string() (в модуле telebot.util), 288
State (класс в telebot.asyncio_ handler_ backends), 282
State (класс в telebot.handler_ backends), 194
state_list() (метод класса telebot.asyncio_ handler_ backends.StatesGroup), 283
state_list() (метод класса telebot.handler_ backends.StatesGroup), 194
StateFilter (класс в telebot.asyncio_ filters), 280
StateFilter (класс в telebot.custom_ filters), 191
StatesGroup (класс в telebot.asyncio_ handler_ backends), 283
StatesGroup (класс в telebot.handler_ backends), 194
Sticker (класс в telebot.types), 90
StickerSet (класс в telebot.types), 91
stop_bot() (метод telebot.TeleBot), 185
stop_message_live_location() (метод telebot.async_ telebot.AsyncTeleBot), 274
stop_message_live_location() (метод telebot.TeleBot), 185
stop_poll() (метод telebot.async_ telebot.AsyncTeleBot), 275
stop_poll() (метод telebot.TeleBot), 186
stop_polling() (метод telebot.TeleBot), 186
Story (класс в telebot.types), 92
SuccessfulPayment (класс в telebot.types), 92
SwitchInlineQueryChosenChat (класс в telebot.types), 93

T

telebot
    module, 103
TeleBot (класс в telebot), 103
telebot.async_telebot
    module, 194
telebot.asyncio_filters
    module, 278
telebot.asyncio_handler_backends
    module, 281
telebot.callback_data
    module, 283
telebot.custom_filters
    module, 189
telebot.formatting
    module, 289
telebot.handler_backends
    module, 193
telebot.types
    module, 4
telebot.util
    module, 284
TextContainsFilter (класс в telebot.asyncio_ filters), 280
TextContainsFilter (класс в telebot.custom_ filters), 191
TextFilter (класс в telebot.asyncio_ filters), 280
TextFilter (класс в telebot.custom_ filters), 192
TextMatchFilter (класс в telebot.asyncio_ filters), 281
TextMatchFilter (класс в telebot.custom_ filters), 192
TextQuote (класс в telebot.types), 93
TextStartsFilter (класс в telebot.asyncio_ filters), 281
TextStartsFilter (класс в telebot.custom_ filters), 192
thumb (telebot.types.Animation property), 4
thumb (telebot.types.Audio property), 5
thumb (telebot.types.Document property), 30
thumb (telebot.types.InputMediaAnimation property), 61
thumb (telebot.types.InputMediaAudio property), 62
thumb (telebot.types.InputMediaDocument property), 63
thumb (telebot.types.InputMediaVideo property), 64
thumb (telebot.types.Sticker property), 91
thumb (telebot.types.StickerSet property), 92
thumb (telebot.types.Video property), 99
thumb (telebot.types.VideoNote property), 100

thumb_height (telebot.types.InlineQueryResultArticle property), 41
thumb_height (telebot.types.InlineQueryResultContact property), 49
thumb_height (telebot.types.InlineQueryResultDocument property), 50
thumb_height (telebot.types.InlineQueryResultLocation property), 52
thumb_height (telebot.types.InlineQueryResultVenue property), 55
thumb_mime_type (telebot.types.InlineQueryResultGif property), 51
thumb_mime_type (telebot.types.InlineQueryResultMpeg4Gif property), 53
thumb_url (telebot.types.InlineQueryResultArticle property), 41
thumb_url (telebot.types.InlineQueryResultContact property), 49
thumb_url (telebot.types.InlineQueryResultDocument property), 50
thumb_url (telebot.types.InlineQueryResultGif property), 51
thumb_url (telebot.types.InlineQueryResultLocation property), 52
thumb_url (telebot.types.InlineQueryResultMpeg4Gif property), 53
thumb_url (telebot.types.InlineQueryResultPhoto property), 54
thumb_url (telebot.types.InlineQueryResultVenue property), 55
thumb_url (telebot.types.InlineQueryResultVideo property), 56
thumb_width (telebot.types.InlineQueryResultArticle property), 41
thumb_width (telebot.types.InlineQueryResultContact property), 49
thumb_width (telebot.types.InlineQueryResultDocument property), 50
thumb_width (telebot.types.InlineQueryResultLocation property), 52
thumb_width (telebot.types.InlineQueryResultVenue property), 55
to_list_of_dicts() (статический метод telebot.types.MessageEntity), 79

U

unban_chat_member() (метод telebot.async_ telebot.AsyncTeleBot), 275
unban_chat_member() (метод telebot.TeleBot), 186
unban_chat_sender_chat() (метод telebot.async_ telebot.AsyncTeleBot), 275
unban_chat_sender_chat() (метод telebot.TeleBot), 187
unhide_general_forum_topic() (метод telebot.async_ telebot.AsyncTeleBot), 276
unhide_general_forum_topic() (метод telebot.TeleBot), 187
unpin_all_chat_messages() (метод telebot.async_ telebot.AsyncTeleBot), 276
unpin_all_chat_messages() (метод telebot.TeleBot), 187
unpin_all_forum_topic_messages() (метод telebot.async_ telebot.AsyncTeleBot), 276
unpin_all_forum_topic_messages() (метод telebot.TeleBot), 188
unpin_all_general_forum_topic_messages() (метод telebot.async_ telebot.AsyncTeleBot), 276
unpin_all_general_forum_topic_messages() (метод telebot.TeleBot), 188
unpin_chat_message() (метод telebot.async_ telebot.AsyncTeleBot), 277
unpin_chat_message() (метод telebot.TeleBot), 188
Update (класс в telebot.types), 94
update_sensitive (атрибут telebot.asyncio_ handler_ backends.BaseMiddleware), 282
update_sensitive (атрибут telebot.handler_ backends.BaseMiddleware), 193
update_types (в модуле telebot.util), 288
upload_sticker_file() (метод telebot.async_ telebot.AsyncTeleBot), 277
upload_sticker_file() (метод telebot.TeleBot), 188
user (telebot.async_ telebot.AsyncTeleBot property), 277
user (telebot.TeleBot property), 189
User (класс в telebot.types), 96
user_id (telebot.types.UsersShared property), 97
user_link() (в модуле telebot.util), 288
user_shared (telebot.types.Message property), 78
UserChatBoosts (класс в telebot.types), 97
UserProfilePhotos (класс в telebot.types), 97
UsersShared (класс в telebot.types), 97

V

validate_web_app_data() (в модуле telebot.util), 289
Venue (класс в telebot.types), 98

Video (*класс в telebot.types*), 98
VideoChatEnded (*класс в telebot.types*), 99
VideoChatParticipantsInvited (*класс в telebot.types*), 99
VideoChatScheduled (*класс в telebot.types*), 99
VideoChatStarted (*класс в telebot.types*), 99
VideoNote (*класс в telebot.types*), 99
Voice (*класс в telebot.types*), 100
voice_chat_ended (*telebot.types.Message property*), 78
voice_chat_participants_invited (*telebot.types.Message property*), 78
voice_chat_scheduled (*telebot.types.Message property*), 78
voice_chat_started (*telebot.types.Message property*), 78
VoiceChatEnded (*класс в telebot.types*), 100
VoiceChatParticipantsInvited (*класс в telebot.types*), 101
VoiceChatScheduled (*класс в telebot.types*), 101
VoiceChatStarted (*класс в telebot.types*), 101

W

WebAppData (*класс в telebot.types*), 101
WebAppInfo (*класс в telebot.types*), 101
webhook_google_functions() (*в модуле telebot.util*), 289
WebhookInfo (*класс в telebot.types*), 101
WriteAccessAllowed (*класс в telebot.types*), 102