

---

# **aiogram Documentation**

***Release 2.11.2***

**Illemius / Alex Root Junior**

**Nov 10, 2020**



# CONTENTS

<b>1</b>	<b>Official aiogram resources</b>	<b>3</b>
<b>2</b>	<b>Features</b>	<b>5</b>
<b>3</b>	<b>Contribute</b>	<b>7</b>
<b>4</b>	<b>Contents</b>	<b>9</b>
4.1	Installation Guide . . . . .	9
4.2	Quick start . . . . .	11
4.3	Migration FAQ (1.4 -> 2.0) . . . . .	12
4.4	Telegram . . . . .	17
4.5	Dispatcher . . . . .	158
4.6	Utils . . . . .	185
4.7	Examples . . . . .	200
4.8	Contribution . . . . .	225
4.9	Links . . . . .	225
<b>5</b>	<b>Indices and tables</b>	<b>227</b>
	<b>Python Module Index</b>	<b>229</b>
	<b>Index</b>	<b>231</b>



**aiogram** is a pretty simple and fully asynchronous framework for [Telegram Bot API](#) written in Python 3.7 with [asyncio](#) and [aiohttp](#). It helps you to make your bots faster and simpler.



## OFFICIAL AIOGRAM RESOURCES

- News: [@aiogram\\_live](#)
- Community: [@aiogram](#)
- Russian community: [@aiogram\\_ru](#)
- Pip: [aiogram](#)
- Docs: [ReadTheDocs](#)
- Source: [Github repo](#)
- Issues/Bug tracker: [Github issues tracker](#)
- Test bot: [@aiogram\\_bot](#)





## FEATURES

- Asynchronous
- Awesome
- Makes things faster
- Has [FSM](#)
- Can reply into webhook. (In other words [make requests in response to updates](#))



**CONTRIBUTE**

- [Issue Tracker](#)
- [Source Code](#)



## CONTENTS

## 4.1 Installation Guide

### 4.1.1 Using PIP

```
$ pip install -U aiogram
```

### 4.1.2 Using Pipenv

```
$ pipenv install aiogram
```

### 4.1.3 Using AUR

*aiogram* is also available in Arch User Repository, so you can install this framework on any Arch-based distribution like ArchLinux, Antergos, Manjaro, etc. To do this, use your favorite AUR-helper and install the [python-aiogram](#) package.

### 4.1.4 From sources

Development versions:

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

Or if you want to install stable version (The same with version from PyPi):

```
$ git clone https://github.com/aiogram/aiogram.git
$ cd aiogram
$ git checkout master
$ python setup.py install
```

### 4.1.5 Recommendations

You can speedup your bots by following next instructions:

- Use `uvloop` instead of default `asyncio` loop.

*uvloop* is a fast, drop-in replacement of the built-in `asyncio` event loop. `uvloop` is implemented in Cython and uses `libuv` under the hood.

**Installation:**

```
$ pip install uvloop
```

- Use `ujson` instead of the default `json` module.

*UltraJSON* is an ultra fast JSON encoder and decoder written in pure C with bindings for Python 2.5+ and 3.

**Installation:**

```
$ pip install ujson
```

- Use `aiohttp` speedups

- Use `cchardet` instead of the `chardet` module.

*cChardet* is a high speed universal character encoding detector.

**Installation:**

```
$ pip install cchardet
```

- Use `aiodns` for speeding up DNS resolving.

*aiodns* provides a simple way for doing asynchronous DNS resolutions.

**Installation:**

```
$ pip install aiodns
```

- Installing speedups altogether.

The following will get you `aiohttp` along with `cchardet`, `aiodns` and `brotlipy` in one bundle.

**Installation:**

```
$ pip install aiohttp[speedups]
```

In addition, you don't need do anything, *aiogram* automatically starts using that if it is found in your environment.

## 4.2 Quick start

### 4.2.1 Simple template

At first you have to import all necessary modules

```
import logging

from aiogram import Bot, Dispatcher, executor, types
```

Then you have to initialize bot and dispatcher instances. Bot token you can get from [@BotFather](#)

```
API_TOKEN = 'BOT TOKEN HERE'

# Configure logging
logging.basicConfig(level=logging.INFO)

# Initialize bot and dispatcher
bot = Bot(token=API_TOKEN)
dp = Dispatcher(bot)
```

Next step: interaction with bots starts with one command. Register your first command handler:

```
@dp.message_handler(commands=['start', 'help'])
async def send_welcome(message: types.Message):
    """
    This handler will be called when user sends `/start` or `/help` command
    """
    await message.reply("Hi!\nI'm EchoBot!\nPowered by aiogram.")
```

If you want to handle all text messages in the chat simply add handler without filters:

```
@dp.message_handler()
async def echo(message: types.Message):
    # old style:
    # await bot.send_message(message.chat.id, message.text)

    await message.answer(message.text)
```

Last step: run long polling.

```
if __name__ == '__main__':
    executor.start_polling(dp, skip_updates=True)
```

### 4.2.2 Summary

```
1 """
2 This is a echo bot.
3 It echoes any incoming text messages.
4 """
5
6 import logging
7
8 from aiogram import Bot, Dispatcher, executor, types
```

(continues on next page)

(continued from previous page)

```

9
10 API_TOKEN = 'BOT TOKEN HERE'
11
12 # Configure logging
13 logging.basicConfig(level=logging.INFO)
14
15 # Initialize bot and dispatcher
16 bot = Bot(token=API_TOKEN)
17 dp = Dispatcher(bot)
18
19
20 @dp.message_handler(commands=['start', 'help'])
21 async def send_welcome(message: types.Message):
22     """
23     This handler will be called when user sends `/start` or `/help` command
24     """
25     await message.reply("Hi!\nI'm EchoBot!\nPowered by aiogram.")
26
27
28
29 @dp.message_handler()
30 async def echo(message: types.Message):
31     # old style:
32     # await bot.send_message(message.chat.id, message.text)
33
34     await message.answer(message.text)
35
36
37 if __name__ == '__main__':
38     executor.start_polling(dp, skip_updates=True)

```

## 4.3 Migration FAQ (1.4 -> 2.0)

This update make breaking changes in aiogram API and drop backward capability with previous versions of framework.

From this point aiogram supports only Python 3.7 and newer.

### 4.3.1 Changelog

- Used contextvars instead of *aiogram.utils.context*;
- Implemented filters factory;
- Implemented new filters mechanism;
- Allowed to customize command prefix in CommandsFilter;
- Implemented mechanism of passing results from filters (as dicts) as kwargs in handlers (like fixtures in pytest);
- Implemented states group feature;
- Implemented FSM storage's proxy;
- Changed files uploading mechanism;
- Implemented pipe for uploading files from URL;



- Implemented I18n Middleware;
- Errors handlers now should accept only two arguments (current update and exception);
- Used *aiohttp\_socks* instead of *aiosocksy* for Socks4/5 proxy;
- *types.ContentType* was divided to *types.ContentType* and *types.ContentTypes*;
- Allowed to use rapidjson instead of ujson/json;
- *.current()* method in bot and dispatcher objects was renamed to *get\_current()*;

## 4.3.2 Instructions

### Contextvars

Context utility (*aiogram.utils.context*) now is removed due to new features of Python 3.7 and all subclasses of *aiogram.types.base.TelegramObject*, *aiogram.Bot* and *aiogram.Dispatcher* has *get\_current()* and *.set\_current()* methods for getting/setting contextual instances of objects.

Example:

```
async def my_handler(message: types.Message):
    bot = Bot.get_current()
    user = types.User.get_current()
    ...
```

### Filters

#### Custom filters

Now *func* keyword argument can't be used for passing filters to the list of filters instead of that you can pass the filters as arguments:

```
@dp.message_handler(lambda message: message.text == 'foo')
@dp.message_handler(types.ChatType.is_private, my_filter)
async def ...
```

(func filter is still available until v2.1)

#### Filters factory

Also you can bind your own filters for using as keyword arguments:

```
from aiogram.dispatcher.filters import BoundFilter

class MyFilter(BoundFilter):
    key = 'is_admin'

    def __init__(self, is_admin):
        self.is_admin = is_admin

    async def check(self, message: types.Message):
        member = await bot.get_chat_member(message.chat.id, message.from_user.id)
        return member.is_chat_admin()
```

(continues on next page)

(continued from previous page)

```
dp.filters_factory.bind(MyFilter)

@dp.message_handler(is_admin=True)
async def ...
```

## Customize commands prefix

Commands prefix can be changed by following one of two available methods:

```
@dp.message_handler(commands=['admin'], commands_prefix='!/')
@dp.message_handler(Command('admin', prefixes='!/'))
async def ...
```

## Passing data from filters as keyword arguments to the handlers

You can pass any data from any filter to the handler by returning `dict`. If any key from the received dictionary not in the handler specification the key will be skipped and will be unavailable from the handler.

Before (<=v1.4)

```
async def my_filter(message: types.Message):
    # do something here
    message.conf['foo'] = 'foo'
    message.conf['bar'] = 42
    return True

@dp.message_handler(func=my_filter)
async def my_message_handler(message: types.Message):
    bar = message.conf["bar"]
    await message.reply(f'bar = {bar}')
```

Now (v2.0)

```
async def my_filter(message: types.Message):
    # do something here
    return {'foo': 'foo', 'bar': 42}

@dp.message_handler(my_filter)
async def my_message_handler(message: types.Message, bar: int):
    await message.reply(f'bar = {bar}')
```

## Other

Filters can also be used as logical expressions:

```
Text(equals='foo') | Text(endswith='Bar') | ~Text(contains='spam')
```

## States group

You can use States objects and States groups instead of string names of the states. String values is still also be available.

Writing states group:

```
from aiogram.dispatcher.filters.state import State, StatesGroup

class UserForm(StatesGroup):
    name = State() # Will be represented in storage as 'Form:name'
    age = State() # Will be represented in storage as 'Form:age'
    gender = State() # Will be represented in storage as 'Form:gender'
```

After that you can use states as *UserForm.name* and etc.

## FSM storage's proxy

Now *Dispatcher.current\_context()* can't be used as context-manager.

Implemented *FSMContext.proxy()* method which returns asynchronous *FSMContextProxy* context manager and can be used for more simply getting data from the storage.

*FSMContextProxy* load all user-related data on initialization and dump it to the storage when proxy is closing if any part of the data was changed.

Usage:

```
@dp.message_handler(commands=['click'])
async def cmd_start(message: types.Message, state: FSMContext):
    async with state.proxy() as proxy: # proxy = FSMContextProxy(state); await proxy.
        ↪load()
        proxy.setdefault('counter', 0)
        proxy['counter'] += 1
        return await message.reply(f"Counter: {proxy['counter']}")
```

This method is not recommended in high-load solutions in reason named “race-condition”.

## File uploading mechanism

Fixed uploading files. Removed *BaseBot.send\_file* method. This allowed to send the *thumb* field.

## Pipe for uploading files from URL

Known issue when Telegram can not accept sending file as URL. In this case need to download file locally and then send.

In this case now you can send file from URL by using pipe. That means you download and send the file without saving it.

You can open the pipe and use for uploading by calling *types.InputFile.from\_file(<URL>)*

Example:

```
URL = 'https://docs.aiogram.dev/en/dev-2.x/_static/logo.png'
```

(continues on next page)

(continued from previous page)

```
@dp.message_handler(commands=['image', 'img'])
async def cmd_image(message: types.Message):
    await bot.send_photo(message.chat.id, types.InputFile.from_url(URL))
```

## I18n Middleware

You can internalize your bot by following next steps:

(Code snippets in this example related with *examples/i18n\_example.py*)

### First usage

1. Extract texts

```
pybabel extract i18n_example.py -o locales/mybot.pot
```

2. Create \*.po files. For e.g. create *en*, *ru*, *uk* locales.
3. Translate texts
4. Compile translations

```
pybabel compile -d locales -D mybot
```

### Updating translations

When you change the code of your bot you need to update *po* & *mo* files:

1. Regenerate pot file:

```
pybabel extract i18n_example.py -o locales/mybot.pot
```

2. Update po files

```
pybabel update -d locales -D mybot -i locales/mybot.pot
```

3. Update your translations
4. Compile *mo* files

```
pybabel compile -d locales -D mybot
```

### Error handlers

Previously errors handlers had to have three arguments *dispatcher*, *update* and *exception* now *dispatcher* argument is removed and will no longer be passed to the error handlers.

## Content types

Content types helper was divided to *types.ContentType* and *types.ContentTypes*.

In filters you can use *types.ContentTypes* but for comparing content types you must use *types.ContentType* class.

## 4.4 Telegram

### 4.4.1 Bot object

#### Low level API

Subclass of this class used only for splitting network interface from all of API methods.

```
class aiogram.bot.base.BaseBot (token: String, loop: Optional[Union[asyncio.base_events.BaseEventLoop,
                                asyncio.events.AbstractEventLoop]] = None, connections_limit: Optional[Integer] = None, proxy: Optional[String] = None,
                                proxy_auth: Optional[aiohttp.helpers.BasicAuth] = None, validate_token: Optional[Boolean] = True, parse_mode: Optional[String] = None,
                                timeout: Optional[Union[Integer, Float, aiohttp.client.ClientTimeout]] = None, server: aiogram.bot.api.TelegramAPIServer = TelegramAPIServer(
                                base='https://api.telegram.org/bot{token}/{method}', file='https://api.telegram.org/file/bot{token}/{path}'))
```

Bases: object

Base class for bot. It's raw bot.

Instructions how to get Bot token is found here: <https://core.telegram.org/bots#3-how-do-i-create-a-bot>

#### Parameters

- **token** (str) – token from @BotFather
- **loop** (Optional Union asyncio.BaseEventLoop, asyncio.AbstractEventLoop) – event loop
- **connections\_limit** (int) – connections limit for aiohttp.ClientSession
- **proxy** (str) – HTTP proxy URL
- **proxy\_auth** (Optional aiohttp.BasicAuth) – Authentication information
- **validate\_token** (bool) – Validate token.
- **parse\_mode** (str) – You can set default parse mode
- **timeout** (typing.Optional[typing.Union[base.Integer, base.Float, aiohttp.ClientTimeout]]) – Request timeout
- **server** (TelegramAPIServer) – Telegram Bot API Server endpoint.

**Raise** when token is invalid throw an *aiogram.utils.exceptions.ValidationError*

**request\_timeout** (timeout: Union[Integer, Float, aiohttp.client.ClientTimeout])

Context manager implements opportunity to change request timeout in current context

**Parameters timeout** (typing.Optional[typing.Union[base.Integer, base.Float, aiohttp.ClientTimeout]]) – Request timeout

**Returns****close()**

Close all client sessions

**async request** (*method: String, data: Optional[Dict] = None, files: Optional[Dict] = None, \*\*kwargs*) → Union[List, Dict, Boolean]

Make an request to Telegram Bot API

<https://core.telegram.org/bots/api#making-requests>**Parameters**

- **method** (str) – API method
- **data** (dict) – request parameters
- **files** (dict) – files

**Returns** result**Return type** Union[List, Dict]**Raise** aiogram.exceptions.TelegramApiError**async download\_file** (*file\_path: String, destination: Optional[InputFile] = None, timeout: Optional[Integer] = <object object>, chunk\_size: Optional[Integer] = 65536, seek: Optional[Boolean] = True*) → Union[\_io.BytesIO, \_io.FileIO]

Download file by file\_path to destination

if You want to automatically create destination (io.BytesIO) use default value of destination and handle result of this method.

**Parameters**

- **file\_path** (str) – file path on telegram server (You can get it from aiogram.types.File)
- **destination** – filename or instance of io.IOBase. For e. g. io.BytesIO
- **timeout** – Integer
- **chunk\_size** – Integer
- **seek** – Boolean - go to start of file when downloading is finished.

**Returns** destination**async send\_file** (*file\_type, method, file, payload*) → Union[Dict, Boolean]

Send file

<https://core.telegram.org/bots/api#inputfile>**Parameters**

- **file\_type** – field name
- **method** – API method
- **file** – String or io.IOBase
- **payload** – request payload

**Returns** response

## Telegram Bot

This class based on `aiogram.bot.base.BaseBot`

```
class aiogram.bot.bot.Bot (token: String, loop: Optional[Union[asyncio.base_events.BaseEventLoop,
    asyncio.events.AbstractEventLoop]] = None, connections_limit: Op-
    tional[Integer] = None, proxy: Optional[String] = None, proxy_auth:
    Optional[aiohttp.helpers.BasicAuth] = None, validate_token: Op-
    tional[Boolean] = True, parse_mode: Optional[String] = None,
    timeout: Optional[Union[Integer, Float, aiohttp.client.ClientTimeout]]
    = None, server: aiogram.bot.api.TelegramAPIServer = Tele-
    gramAPIServer(base='https://api.telegram.org/bot{token}/{method}',
    file='https://api.telegram.org/file/bot{token}/{path}'))

Bases: aiogram.bot.base.BaseBot, aiogram.utils.mixins.DataMixin, aiogram.
utils.mixins.ContextInstanceMixin
```

Base bot class

Instructions how to get Bot token is found here: <https://core.telegram.org/bots#3-how-do-i-create-a-bot>

### Parameters

- **token** (str) – token from @BotFather
- **loop** (Optional Union asyncio.BaseEventLoop, asyncio.AbstractEventLoop) – event loop
- **connections\_limit** (int) – connections limit for aiohttp.ClientSession
- **proxy** (str) – HTTP proxy URL
- **proxy\_auth** (Optional aiohttp.BasicAuth) – Authentication information
- **validate\_token** (bool) – Validate token.
- **parse\_mode** (str) – You can set default parse mode
- **timeout** (typing.Optional[typing.Union[base.Integer, base.Float, aiohttp.ClientTimeout]]) – Request timeout
- **server** (TelegramAPIServer) – Telegram Bot API Server endpoint.

**Raise** when token is invalid throw an `aiogram.utils.exceptions.ValidationError`

### property me

Alias for self.get\_me() but lazy and with caching.

**Returns** `aiogram.types.User`

```
async download_file_by_id (file_id: String, destination=None, timeout: Integer = 30,
    chunk_size: Integer = 65536, seek: Boolean = True)
```

Download file by file\_id to destination

if You want to automatically create destination (`io.BytesIO`) use default value of destination and handle result of this method.

### Parameters

- **file\_id** – str
- **destination** – filename or instance of `io.IOBase`. For e. g. `io.BytesIO`
- **timeout** – int
- **chunk\_size** – int

- **seek** – bool - go to start of file when downloading is finished

**Returns** destination

**async get\_updates** (*offset: Optional[Integer] = None, limit: Optional[Integer] = None, timeout: Optional[Integer] = None, allowed\_updates: Optional[List[String]] = None*)  
→ List[aiogram.types.update.Update]

Use this method to receive incoming updates using long polling (wiki).

Notes 1. This method will not work if an outgoing webhook is set up. 2. In order to avoid getting duplicate updates, recalculate offset after each server response.

Source: <https://core.telegram.org/bots/api#getupdates>

#### Parameters

- **offset** (typing.Optional[base.Integer]) – Identifier of the first update to be returned
- **limit** (typing.Optional[base.Integer]) – Limits the number of updates to be retrieved
- **timeout** (typing.Optional[base.Integer]) – Timeout in seconds for long polling
- **allowed\_updates** (typing.Union[typing.List[base.String], None]) – List the types of updates you want your bot to receive

**Returns** An Array of Update objects is returned

**Return type** typing.List[types.Update]

**async set\_webhook** (*url: String, certificate: Optional[InputFile] = None, ip\_address: Optional[String] = None, max\_connections: Optional[Integer] = None, allowed\_updates: Optional[List[String]] = None, drop\_pending\_updates: Optional[Boolean] = None*) → Boolean

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

If you'd like to make sure that the Webhook request comes from Telegram, we recommend using a secret path in the URL, e.g. <https://www.example.com/<token>>. Since nobody else knows your bot's token, you can be pretty sure it's us.

Source: <https://core.telegram.org/bots/api#setwebhook>

#### Parameters

- **url** (base.String) – HTTPS url to send updates to. Use an empty string to remove webhook integration
- **certificate** (typing.Optional[base.InputFile]) – Upload your public key certificate so that the root certificate in use can be checked. See our self-signed guide for details: <https://core.telegram.org/bots/self-signed>
- **ip\_address** (typing.Optional[base.String]) – The fixed IP address which will be used to send webhook requests instead of the IP address resolved through DNS
- **max\_connections** (typing.Optional[base.Integer]) – Maximum allowed number of simultaneous HTTPS connections to the webhook for update delivery, 1-100. Defaults to 40. Use lower values to limit the load on your bot's server, and higher values to increase your bot's throughput.



- **allowed\_updates** (`typing.Optional[typing.List[base.String]]`) – A list of the update types you want your bot to receive. For example, specify [“message”, “edited\_channel\_post”, “callback\_query”] to only receive updates of these types. See Update for a complete list of available update types. Specify an empty list to receive all updates regardless of type (default). If not specified, the previous setting will be used. Please note that this parameter doesn’t affect updates created before the call to the `setWebhook`, so unwanted updates may be received for a short period of time.
- **drop\_pending\_updates** (`typing.Optional[base.Boolean]`) – Pass `True` to drop all pending updates

**Returns** Returns `true`

**Return type** `base.Boolean`

**async delete\_webhook** (*drop\_pending\_updates: Optional[Boolean] = None*) → `Boolean`

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

Source: <https://core.telegram.org/bots/api#deletewebhook>

**Parameters** **drop\_pending\_updates** (`typing.Optional[base.Boolean]`) – Pass `True` to drop all pending updates

**Returns** Returns `True` on success

**Return type** `base.Boolean`

**async get\_webhook\_info** () → *aiogram.types.webhook\_info.WebhookInfo*

Use this method to get current webhook status. Requires no parameters.

If the bot is using `getUpdates`, will return an object with the `url` field empty.

Source: <https://core.telegram.org/bots/api#getwebhookinfo>

**Returns** On success, returns a `WebhookInfo` object

**Return type** `types.WebhookInfo`

**async get\_me** () → *aiogram.types.user.User*

A simple method for testing your bot’s auth token. Requires no parameters.

Source: <https://core.telegram.org/bots/api#getme>

**Returns** Returns basic information about the bot in form of a `User` object

**Return type** `types.User`

**async log\_out** () → `Boolean`

Use this method to log out from the cloud Bot API server before launching the bot locally. You **must** log out the bot before running it locally, otherwise there is no guarantee that the bot will receive updates. After a successful call, you will not be able to log in again using the same token for 10 minutes. Returns `True` on success. Requires no parameters.

Source: <https://core.telegram.org/bots/api#logout>

**Returns** Returns `True` on success

**Return type** `base.Boolean`

**close\_bot** () → `Boolean`

Use this method to close the bot instance before moving it from one local server to another. You need to delete the webhook before calling this method to ensure that the bot isn’t launched again after server restart. The method will return error 429 in the first 10 minutes after the bot is launched. Returns `True` on success. Requires no parameters.

Source: <https://core.telegram.org/bots/api#close>

**Returns** Returns True on success

**Return type** `base.Boolean`

```
async send_message (chat_id: Union[Integer, String], text: String,
                    parse_mode: Optional[String] = None, entities: Optional[List[aiogram.types.message_entity.MessageEntity]] =
                    None, disable_web_page_preview: Optional[Boolean] =
                    None, disable_notification: Optional[Boolean] = None,
                    reply_to_message_id: Optional[Integer] = None,
                    allow_sending_without_reply: Optional[Boolean] = None, reply_markup:
                    Optional[Union[aiogram.types.inline_keyboard.InlineKeyboardMarkup,
                    aiogram.types.reply_keyboard.ReplyKeyboardMarkup,
                    aiogram.types.reply_keyboard.ReplyKeyboardRemove,
                    aiogram.types.force_reply.ForceReply]] = None) →
                    aiogram.types.message.Message
```

Use this method to send text messages.

Source: <https://core.telegram.org/bots/api#sendmessage>

#### Parameters

- **chat\_id** (`typing.Union[base.Integer, base.String]`) – Unique identifier for the target chat or username of the target channel
- **text** (`base.String`) – Text of the message to be sent
- **parse\_mode** (`typing.Optional[base.String]`) – Send Markdown or HTML, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in your bot's message.
- **entities** (`typing.Optional[typing.List[types.MessageEntity]]`) – List of special entities that appear in message text, which can be specified instead of `parse_mode`
- **disable\_web\_page\_preview** (`typing.Optional[base.Boolean]`) – Disables link previews for links in this message
- **disable\_notification** (`typing.Optional[base.Boolean]`) – Sends the message silently. Users will receive a notification with no sound
- **reply\_to\_message\_id** (`typing.Optional[base.Integer]`) – If the message is a reply, ID of the original message
- **allow\_sending\_without\_reply** (`typing.Optional[base.Boolean]`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- **reply\_markup** (`typing.Union[types.InlineKeyboardMarkup, types.ReplyKeyboardMarkup, types.ReplyKeyboardRemove, types.ForceReply, None]`) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user

**Returns** On success, the sent Message is returned

**Return type** `types.Message`

```
async forward_message (chat_id: Union[Integer, String], from_chat_id: Union[Integer, String],
                        message_id: Integer, disable_notification: Optional[Boolean] = None)
                        → aiogram.types.message.Message
```

Use this method to forward messages of any kind.

Source: <https://core.telegram.org/bots/api#forwardmessage>

#### Parameters

- **chat\_id** (typing.Union[base.Integer, base.String]) – Unique identifier for the target chat or username of the target channel
- **from\_chat\_id** (typing.Union[base.Integer, base.String]) – Unique identifier for the chat where the original message was sent
- **disable\_notification** (typing.Optional[base.Boolean]) – Sends the message silently. Users will receive a notification with no sound
- **message\_id** (base.Integer) – Message identifier in the chat specified in from\_chat\_id

**Returns** On success, the sent Message is returned

**Return type** types.Message

```
async copy_message (chat_id: Union[Integer, String], from_chat_id: Union[Integer,
String], message_id: Integer, caption: Optional[String] =
None, parse_mode: Optional[String] = None, caption_entities:
Optional[List[aiogram.types.message_entity.MessageEntity]]
= None, disable_notification: Optional[Boolean] = None,
reply_to_message_id: Optional[Integer] = None, al-
low_sending_without_reply: Optional[Boolean] = None, reply_markup:
Optional[Union[aiogram.types.inline_keyboard.InlineKeyboardMarkup,
aiogram.types.reply_keyboard.ReplyKeyboardMarkup,
aiogram.types.reply_keyboard.ReplyKeyboardRemove,
aiogram.types.force_reply.ForceReply]] = None) →
aiogram.types.message_id.MessageId
```

Use this method to copy messages of any kind. The method is analogous to the method forwardMessages, but the copied message doesn't have a link to the original message. Returns the MessageId of the sent message on success.

Source: <https://core.telegram.org/bots/api#copymessage>

#### Parameters

- **chat\_id** (typing.Union[base.Integer, base.String]) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- **from\_chat\_id** (typing.Union[base.Integer, base.String]) – Unique identifier for the chat where the original message was sent (or channel username in the format @channelusername)
- **message\_id** (base.Integer) – Message identifier in the chat specified in from\_chat\_id
- **caption** (typing.Optional[base.String]) – New caption for media, 0-1024 characters after entities parsing. If not specified, the original caption is kept
- **parse\_mode** (typing.Optional[base.String]) – Mode for parsing entities in the new caption. See formatting options for more details: <https://core.telegram.org/bots/api#formatting-options>

- **caption\_entities** (`typing.Optional[typing.List[types.MessageEntity]]`) – List of special entities that appear in the new caption, which can be specified instead of `parse_mode`
- **disable\_notification** (`typing.Optional[base.Boolean]`) – Sends the message silently. Users will receive a notification with no sound
- **reply\_to\_message\_id** (`typing.Optional[base.Integer]`) – If the message is a reply, ID of the original message
- **allow\_sending\_without\_reply** (`typing.Optional[base.Boolean]`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- **reply\_markup** (`typing.Union[types.InlineKeyboardMarkup, types.ReplyKeyboardMarkup, types.ReplyKeyboardRemove, types.ForceReply, None]`) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user.

**Returns** On success, the sent Message is returned

**Return type** `types.Message`

```
async send_photo(chat_id: Union[Integer, String], photo: Union[InputFile, String], caption:
Optional[String] = None, parse_mode: Optional[String] = None, cap-
tion_entities: Optional[List[aiogram.types.message_entity.MessageEntity]]
= None, disable_notification: Optional[Boolean] = None,
reply_to_message_id: Optional[Integer] = None, al-
low_sending_without_reply: Optional[Boolean] = None, reply_markup:
Optional[Union[aiogram.types.inline_keyboard.InlineKeyboardMarkup,
aiogram.types.reply_keyboard.ReplyKeyboardMarkup,
aiogram.types.reply_keyboard.ReplyKeyboardRemove,
aiogram.types.force_reply.ForceReply]] = None) →
aiogram.types.message.Message
```

Use this method to send photos.

Source: <https://core.telegram.org/bots/api#sendphoto>

#### Parameters

- **chat\_id** (`typing.Union[base.Integer, base.String]`) – Unique identifier for the target chat or username of the target channel
- **photo** (`typing.Union[base.InputFile, base.String]`) – Photo to send
- **caption** (`typing.Optional[base.String]`) – Photo caption (may also be used when resending photos by `file_id`), 0-1024 characters
- **parse\_mode** (`typing.Optional[base.String]`) – Send Markdown or HTML, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in your bot's message.
- **caption\_entities** (`typing.Optional[typing.List[types.MessageEntity]]`) – List of special entities that appear in message text, which can be specified instead of `parse_mode`
- **disable\_notification** (`typing.Optional[base.Boolean]`) – Sends the message silently. Users will receive a notification with no sound
- **reply\_to\_message\_id** (`typing.Optional[base.Integer]`) – If the message is a reply, ID of the original message

- **allow\_sending\_without\_reply** (`typing.Optional[base.Boolean]`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- **reply\_markup** (`typing.Union[types.InlineKeyboardMarkup, types.ReplyKeyboardMarkup, types.ReplyKeyboardRemove, types.ForceReply, None]`) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user

**Returns** On success, the sent Message is returned

**Return type** `types.Message`

```
async send_audio(chat_id: Union[Integer, String], audio: Union[InputFile, String], caption:
Optional[String] = None, parse_mode: Optional[String] = None, cap-
tion_entities: Optional[List[aiogram.types.message_entity.MessageEntity]]
= None, duration: Optional[Integer] = None, performer: Op-
tional[String] = None, title: Optional[String] = None, thumb: Op-
tional[Union[InputFile, String]] = None, disable_notification: Op-
tional[Boolean] = None, reply_to_message_id: Optional[Integer] = None,
allow_sending_without_reply: Optional[Boolean] = None, reply_markup:
Optional[Union[aiogram.types.inline_keyboard.InlineKeyboardMarkup,
aiogram.types.reply_keyboard.ReplyKeyboardMarkup,
aiogram.types.reply_keyboard.ReplyKeyboardRemove,
aiogram.types.force_reply.ForceReply]] = None) →
aiogram.types.message.Message
```

Use this method to send audio files, if you want Telegram clients to display them in the music player. Your audio must be in the .mp3 format.

For sending voice messages, use the `sendVoice` method instead.

Source: <https://core.telegram.org/bots/api#sendaudio>

#### Parameters

- **chat\_id** (`typing.Union[base.Integer, base.String]`) – Unique identifier for the target chat or username of the target channel
- **audio** (`typing.Union[base.InputFile, base.String]`) – Audio file to send
- **caption** (`typing.Optional[base.String]`) – Audio caption, 0-1024 characters
- **parse\_mode** (`typing.Optional[base.String]`) – Send Markdown or HTML, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in your bot's message.
- **caption\_entities** (`typing.Optional[typing.List[types.MessageEntity]]`) – List of special entities that appear in message text, which can be specified instead of `parse_mode`
- **duration** (`typing.Optional[base.Integer]`) – Duration of the audio in seconds
- **performer** (`typing.Optional[base.String]`) – Performer
- **title** (`typing.Optional[base.String]`) – Track name
- **thumb** (`typing.Union[base.InputFile, base.String, None]`) – Thumbnail of the file sent

- **disable\_notification** (`typing.Optional[base.Boolean]`) – Sends the message silently. Users will receive a notification with no sound
- **reply\_to\_message\_id** (`typing.Optional[base.Integer]`) – If the message is a reply, ID of the original message
- **allow\_sending\_without\_reply** (`typing.Optional[base.Boolean]`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- **reply\_markup** (`typing.Union[types.InlineKeyboardMarkup, types.ReplyKeyboardMarkup, types.ReplyKeyboardRemove, types.ForceReply, None]`) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user

**Returns** On success, the sent Message is returned

**Return type** `types.Message`

```
async send_document (chat_id: Union[Integer, String], document: Union[InputFile, String], thumb: Optional[Union[InputFile, String]] = None, caption: Optional[String] = None, parse_mode: Optional[String] = None, caption_entities: Optional[List[aiogram.types.message_entity.MessageEntity]] = None, disable_content_type_detection: Optional[Boolean] = None, disable_notification: Optional[Boolean] = None, reply_to_message_id: Optional[Integer] = None, allow_sending_without_reply: Optional[Boolean] = None, reply_markup: Optional[Union[aiogram.types.inline_keyboard.InlineKeyboardMarkup, aiogram.types.reply_keyboard.ReplyKeyboardMarkup, aiogram.types.reply_keyboard.ReplyKeyboardRemove, aiogram.types.force_reply.ForceReply]] = None) → aiogram.types.message.Message
```

Use this method to send general files. On success, the sent Message is returned. Bots can currently send files of any type of up to 50 MB in size, this limit may be changed in the future.

Source: <https://core.telegram.org/bots/api#senddocument>

#### Parameters

- **chat\_id** (`typing.Union[base.Integer, base.String]`) – Unique identifier for the target chat or username of the target channel
- **document** (`typing.Union[base.InputFile, base.String]`) – File to send
- **thumb** (`typing.Union[base.InputFile, base.String, None]`) – Thumbnail of the file sent
- **caption** (`typing.Optional[base.String]`) – Document caption (may also be used when resending documents by file\_id), 0-1024 characters
- **disable\_content\_type\_detection** (`typing.Optional[base.Boolean]`) – Disables automatic server-side content type detection for files uploaded using multipart/form-data
- **parse\_mode** (`typing.Optional[base.String]`) – Send Markdown or HTML, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in your bot's message.

- **caption\_entities** (`typing.Optional[typing.List[types.MessageEntity]]`) – List of special entities that appear in message text, which can be specified instead of `parse_mode`
- **disable\_notification** (`typing.Optional[base.Boolean]`) – Sends the message silently. Users will receive a notification with no sound
- **reply\_to\_message\_id** (`typing.Optional[base.Integer]`) – If the message is a reply, ID of the original message
- **allow\_sending\_without\_reply** (`typing.Optional[base.Boolean]`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- **reply\_markup** (`typing.Union[types.InlineKeyboardMarkup, types.ReplyKeyboardMarkup, types.ReplyKeyboardRemove, types.ForceReply, None]`) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user

**Returns** On success, the sent Message is returned

**Return type** `types.Message`

```
async send_video(chat_id: Union[Integer, String], video: Union[InputFile, String], duration:
    Optional[Integer] = None, width: Optional[Integer] = None, height: Op-
    tional[Integer] = None, thumb: Optional[Union[InputFile, String]] = None,
    caption: Optional[String] = None, parse_mode: Optional[String] = None,
    caption_entities: Optional[List[aiogram.types.message_entity.MessageEntity]]
    = None, supports_streaming: Optional[Boolean] = None, disable_notification:
    Optional[Boolean] = None, reply_to_message_id: Optional[Integer] = None,
    allow_sending_without_reply: Optional[Boolean] = None, reply_markup:
    Optional[Union[aiogram.types.inline_keyboard.InlineKeyboardMarkup,
    aiogram.types.reply_keyboard.ReplyKeyboardMarkup,
    aiogram.types.reply_keyboard.ReplyKeyboardRemove,
    aiogram.types.force_reply.ForceReply]] = None) →
    aiogram.types.message.Message
```

Use this method to send video files, Telegram clients support mp4 videos (other formats may be sent as Document).

Source: <https://core.telegram.org/bots/api#sendvideo>

#### Parameters

- **chat\_id** (`typing.Union[base.Integer, base.String]`) – Unique identifier for the target chat or username of the target channel
- **video** (`typing.Union[base.InputFile, base.String]`) – Video to send
- **duration** (`typing.Optional[base.Integer]`) – Duration of sent video in seconds
- **width** (`typing.Optional[base.Integer]`) – Video width
- **height** (`typing.Optional[base.Integer]`) – Video height
- **thumb** (`typing.Union[base.InputFile, base.String, None]`) – Thumbnail of the file sent
- **caption** (`typing.Optional[base.String]`) – Video caption (may also be used when resending videos by `file_id`), 0-1024 characters



- **parse\_mode** (`typing.Optional[base.String]`) – Send Markdown or HTML, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in your bot's message.
- **caption\_entities** (`typing.Optional[typing.List[types.MessageEntity]]`) – List of special entities that appear in message text, which can be specified instead of `parse_mode`
- **supports\_streaming** (`typing.Optional[base.Boolean]`) – Pass True, if the uploaded video is suitable for streaming
- **disable\_notification** (`typing.Optional[base.Boolean]`) – Sends the message silently. Users will receive a notification with no sound
- **reply\_to\_message\_id** (`typing.Optional[base.Integer]`) – If the message is a reply, ID of the original message
- **allow\_sending\_without\_reply** (`typing.Optional[base.Boolean]`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- **reply\_markup** (`typing.Union[types.InlineKeyboardMarkup, types.ReplyKeyboardMarkup, types.ReplyKeyboardRemove, types.ForceReply, None]`) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user

**Returns** On success, the sent Message is returned

**Return type** `types.Message`

```
async send_animation (chat_id: Union[Integer, String], animation: Union[InputFile, String], duration: Optional[Integer] = None, width: Optional[Integer] = None, height: Optional[Integer] = None, thumb: Optional[Union[InputFile, String]] = None, caption: Optional[String] = None, parse_mode: Optional[String] = None, caption_entities: Optional[List[aiogram.types.message_entity.MessageEntity]] = None, disable_notification: Optional[Boolean] = None, reply_to_message_id: Optional[Integer] = None, allow_sending_without_reply: Optional[Boolean] = None, reply_markup: Optional[Union[aiogram.types.inline_keyboard.InlineKeyboardMarkup, aiogram.types.reply_keyboard.ReplyKeyboardMarkup, aiogram.types.reply_keyboard.ReplyKeyboardRemove, aiogram.types.force_reply.ForceReply]] = None) → aiogram.types.message.Message
```

Use this method to send animation files (GIF or H.264/MPEG-4 AVC video without sound).

On success, the sent Message is returned. Bots can currently send animation files of up to 50 MB in size, this limit may be changed in the future.

Source <https://core.telegram.org/bots/api#sendanimation>

#### Parameters

- **chat\_id** (`typing.Union[base.Integer, base.String]`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- **animation** (`typing.Union[base.InputFile, base.String]`) – Animation to send. Pass a `file_id` as String to send an animation that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get an animation from the Internet, or upload a new animation using multipart/form-data



- **duration** (`typing.Optional[base.Integer]`) – Duration of sent animation in seconds
- **width** (`typing.Optional[base.Integer]`) – Animation width
- **height** (`typing.Optional[base.Integer]`) – Animation height
- **thumb** (`typing.Union[typing.Union[base.InputFile, base.String], None]`) – Thumbnail of the file sent. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320.
- **caption** (`typing.Optional[base.String]`) – Animation caption (may also be used when resending animation by `file_id`), 0-1024 characters
- **parse\_mode** (`typing.Optional[base.String]`) – Send Markdown or HTML, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in the media caption
- **caption\_entities** (`typing.Optional[typing.List[types.MessageEntity]]`) – List of special entities that appear in message text, which can be specified instead of `parse_mode`
- **disable\_notification** (`typing.Optional[base.Boolean]`) – Sends the message silently. Users will receive a notification with no sound
- **reply\_to\_message\_id** (`typing.Optional[base.Integer]`) – If the message is a reply, ID of the original message
- **allow\_sending\_without\_reply** (`typing.Optional[base.Boolean]`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- **reply\_markup** (`typing.Union[typing.Union[types.InlineKeyboardMarkup, types.ReplyKeyboardMarkup, types.ReplyKeyboardRemove, types.ForceReply], None]`) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user

**Returns** On success, the sent Message is returned

**Return type** `types.Message`

```
async send_voice(chat_id: Union[Integer, String], voice: Union[InputFile, String], caption:
Optional[String] = None, parse_mode: Optional[String] = None, cap-
tion_entities: Optional[List[aiogram.types.message_entity.MessageEntity]]
= None, duration: Optional[Integer] = None, disable_notification: Op-
tional[Boolean] = None, reply_to_message_id: Optional[Integer] = None,
allow_sending_without_reply: Optional[Boolean] = None, reply_markup:
Optional[Union[aiogram.types.inline_keyboard.InlineKeyboardMarkup,
aiogram.types.reply_keyboard.ReplyKeyboardMarkup,
aiogram.types.reply_keyboard.ReplyKeyboardRemove,
aiogram.types.force_reply.ForceReply]] = None) →
aiogram.types.message.Message
```

Use this method to send audio files, if you want Telegram clients to display the file as a playable voice message.

For this to work, your audio must be in an .ogg file encoded with OPUS (other formats may be sent as Audio or Document).

Source: <https://core.telegram.org/bots/api#sendvoice>

### Parameters

- **chat\_id** (`typing.Union[base.Integer, base.String]`) – Unique identifier for the target chat or username of the target channel
- **voice** (`typing.Union[base.InputFile, base.String]`) – Audio file to send
- **caption** (`typing.Optional[base.String]`) – Voice message caption, 0-1024 characters
- **parse\_mode** (`typing.Optional[base.String]`) – Send Markdown or HTML, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in your bot's message.
- **caption\_entities** (`typing.Optional[typing.List[types.MessageEntity]]`) – List of special entities that appear in message text, which can be specified instead of `parse_mode`
- **duration** (`typing.Optional[base.Integer]`) – Duration of the voice message in seconds
- **disable\_notification** (`typing.Optional[base.Boolean]`) – Sends the message silently. Users will receive a notification with no sound
- **reply\_to\_message\_id** (`typing.Optional[base.Integer]`) – If the message is a reply, ID of the original message
- **allow\_sending\_without\_reply** (`typing.Optional[base.Boolean]`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- **reply\_markup** (`typing.Union[types.InlineKeyboardMarkup, types.ReplyKeyboardMarkup, types.ReplyKeyboardRemove, types.ForceReply, None]`) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user

**Returns** On success, the sent Message is returned

**Return type** `types.Message`

```
async send_video_note (chat_id: Union[Integer, String], video_note: Union[InputFile, String],
                        duration: Optional[Integer] = None, length: Optional[Integer] =
                        None, thumb: Optional[Union[InputFile, String]] = None, dis-
                        able_notification: Optional[Boolean] = None, reply_to_message_id:
                        Optional[Integer] = None, allow_sending_without_reply:
                        Optional[Boolean] = None, reply_markup: Op-
                        tional[Union[aiogram.types.inline_keyboard.InlineKeyboardMarkup,
                        aiogram.types.reply_keyboard.ReplyKeyboardMarkup,
                        aiogram.types.reply_keyboard.ReplyKeyboardRemove,
                        aiogram.types.force_reply.ForceReply]] = None) →
                        aiogram.types.message.Message
```

As of v.4.0, Telegram clients support rounded square mp4 videos of up to 1 minute long. Use this method to send video messages.

Source: <https://core.telegram.org/bots/api#sendvideonote>

### Parameters

- **chat\_id** (`typing.Union[base.Integer, base.String]`) – Unique identifier for the target chat or username of the target channel

- **video\_note** (`typing.Union[base.InputFile, base.String]`) – Video note to send
- **duration** (`typing.Optional[base.Integer]`) – Duration of sent video in seconds
- **length** (`typing.Optional[base.Integer]`) – Video width and height
- **thumb** (`typing.Union[base.InputFile, base.String, None]`) – Thumbnail of the file sent
- **disable\_notification** (`typing.Optional[base.Boolean]`) – Sends the message silently. Users will receive a notification with no sound
- **reply\_to\_message\_id** (`typing.Optional[base.Integer]`) – If the message is a reply, ID of the original message
- **allow\_sending\_without\_reply** (`typing.Optional[base.Boolean]`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- **reply\_markup** (`typing.Union[types.InlineKeyboardMarkup, types.ReplyKeyboardMarkup, types.ReplyKeyboardRemove, types.ForceReply, None]`) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user

**Returns** On success, the sent Message is returned

**Return type** `types.Message`

```
async send_media_group (chat_id: Union[Integer, String], media: Union[aiogram.types.input_media.MediaGroup, List], disable_notification: Optional[Boolean] = None, reply_to_message_id: Optional[Integer] = None, allow_sending_without_reply: Optional[Boolean] = None) → List[aiogram.types.message.Message]
```

Use this method to send a group of photos, videos, documents or audios as an album. Documents and audio files can be only group in an album with messages of the same type. On success, an array of Messages that were sent is returned.

Source: <https://core.telegram.org/bots/api#sendmediagroup>

#### Parameters

- **chat\_id** (`typing.Union[base.Integer, base.String]`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- **media** (`typing.Union[types.MediaGroup, typing.List]`) – A JSON-serialized array describing messages to be sent, must include 2-10 items
- **disable\_notification** (`typing.Optional[base.Boolean]`) – Sends messages silently. Users will receive a notification with no sound.
- **reply\_to\_message\_id** (`typing.Optional[base.Integer]`) – If the messages are a reply, ID of the original message
- **allow\_sending\_without\_reply** (`typing.Optional[base.Boolean]`) – Pass True, if the message should be sent even if the specified replied-to message is not found

**Returns** On success, an array of the sent Messages is returned

**Return type** `typing.List[types.Message]`

```
async send_location(chat_id: Union[Integer, String], latitude: Float, longitude: Float,
horizontal_accuracy: Optional[Float] = None, live_period: Optional[Integer] = None, heading: Optional[Integer] = None, proximity_alert_radius: Optional[Integer] = None, disable_notification: Optional[Boolean] = None, reply_to_message_id: Optional[Integer] = None, allow_sending_without_reply: Optional[Boolean] = None, reply_markup: Optional[Union[aiogram.types.inline_keyboard.InlineKeyboardMarkup, aiogram.types.reply_keyboard.ReplyKeyboardMarkup, aiogram.types.reply_keyboard.ReplyKeyboardRemove, aiogram.types.force_reply.ForceReply]] = None) → aiogram.types.message.Message
```

Use this method to send point on the map.

Source: <https://core.telegram.org/bots/api#sendlocation>

#### Parameters

- **chat\_id** (`typing.Union[base.Integer, base.String]`) – Unique identifier for the target chat or username of the target channel
- **latitude** (`base.Float`) – Latitude of the location
- **longitude** (`base.Float`) – Longitude of the location
- **horizontal\_accuracy** (`typing.Optional[base.Float]`) – The radius of uncertainty for the location, measured in meters; 0-1500
- **live\_period** (`typing.Optional[base.Integer]`) – Period in seconds for which the location will be updated
- **heading** (`typing.Optional[base.Integer]`) – For live locations, a direction in which the user is moving, in degrees. Must be between 1 and 360 if specified.
- **proximity\_alert\_radius** (`typing.Optional[base.Integer]`) – For live locations, a maximum distance for proximity alerts about approaching another chat member, in meters. Must be between 1 and 100000 if specified.
- **disable\_notification** (`typing.Optional[base.Boolean]`) – Sends the message silently. Users will receive a notification with no sound
- **reply\_to\_message\_id** (`typing.Optional[base.Integer]`) – If the message is a reply, ID of the original message
- **allow\_sending\_without\_reply** (`typing.Optional[base.Boolean]`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- **reply\_markup** (`typing.Union[types.InlineKeyboardMarkup, types.ReplyKeyboardMarkup, types.ReplyKeyboardRemove, types.ForceReply, None]`) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user

**Returns** On success, the sent Message is returned

**Return type** `types.Message`

```
async edit_message_live_location (latitude: Float, longitude: Float, chat_id:
Optional[Union[Integer, String]] = None,
message_id: Optional[Integer] = None, in-
line_message_id: Optional[String] = None, hori-
zontal_accuracy: Optional[Float] = None, heading:
Optional[Integer] = None, proximity_alert_radius:
Optional[Integer] = None, reply_markup: Op-
tional[aiogram.types.inline_keyboard.InlineKeyboardMarkup]
= None) → aiogram.types.message.Message
```

Use this method to edit live location messages sent by the bot or via the bot (for inline bots). A location can be edited until its live\_period expires or editing is explicitly disabled by a call to stopMessageLiveLocation.

Source: <https://core.telegram.org/bots/api#editmessagelivelocation>

#### Parameters

- **chat\_id** (typing.Union[base.Integer, base.String, None]) – Required if inline\_message\_id is not specified
- **message\_id** (typing.Optional[base.Integer]) – Required if inline\_message\_id is not specified. Identifier of the sent message
- **inline\_message\_id** (typing.Optional[base.String]) – Required if chat\_id and message\_id are not specified. Identifier of the inline message
- **latitude** (base.Float) – Latitude of new location
- **longitude** (base.Float) – Longitude of new location
- **horizontal\_accuracy** (typing.Optional[base.Float]) – The radius of uncertainty for the location, measured in meters; 0-1500
- **heading** (typing.Optional[base.Integer]) – Direction in which the user is moving, in degrees. Must be between 1 and 360 if specified.
- **proximity\_alert\_radius** (typing.Optional[base.Integer]) – 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** (typing.Optional[types.InlineKeyboardMarkup]) – A JSON-serialized object for a new inline keyboard

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

**Return type** typing.Union[types.Message, base.Boolean]

```
async stop_message_live_location (chat_id: Optional[Union[Integer, String]] = None, mes-
sage_id: Optional[Integer] = None, inline_message_id:
Optional[String] = None, reply_markup: Op-
tional[aiogram.types.inline_keyboard.InlineKeyboardMarkup]
= None) → aiogram.types.message.Message
```

Use this method to stop updating a live location message sent by the bot or via the bot (for inline bots) before live\_period expires.

Source: <https://core.telegram.org/bots/api#stopmessagelivelocation>

#### Parameters

- **chat\_id** (typing.Union[base.Integer, base.String, None]) – Required if inline\_message\_id is not specified

- **message\_id** (`typing.Optional[base.Integer]`) – Required if `inline_message_id` is not specified. Identifier of the sent message
- **inline\_message\_id** (`typing.Optional[base.String]`) – Required if `chat_id` and `message_id` are not specified. Identifier of the inline message
- **reply\_markup** (`typing.Optional[types.InlineKeyboardMarkup]`) – A JSON-serialized object for a new inline keyboard

**Returns** On success, if the message was sent by the bot, the sent `Message` is returned, otherwise `True` is returned.

**Return type** `typing.Union[types.Message, base.Boolean]`

```
async send_venue (chat_id: Union[Integer, String], latitude: Float, longitude: Float, title: String,
                    address: String, foursquare_id: Optional[String] = None, foursquare_type:
                    Optional[String] = None, google_place_id: Optional[String] = None,
                    google_place_type: Optional[String] = None, disable_notification: Op-
                    tional[Boolean] = None, reply_to_message_id: Optional[Integer] = None,
                    allow_sending_without_reply: Optional[Boolean] = None, reply_markup:
                    Optional[Union[aiogram.types.inline_keyboard.InlineKeyboardMarkup,
                    aiogram.types.reply_keyboard.ReplyKeyboardMarkup,
                    aiogram.types.reply_keyboard.ReplyKeyboardRemove,
                    aiogram.types.force_reply.ForceReply]] = None) →
                    aiogram.types.message.Message
```

Use this method to send information about a venue.

Source: <https://core.telegram.org/bots/api#sendvenue>

#### Parameters

- **chat\_id** (`typing.Union[base.Integer, base.String]`) – Unique identifier for the target chat or username of the target channel (in the format `@channelusername`)
- **latitude** (`base.Float`) – Latitude of the venue
- **longitude** (`base.Float`) – Longitude of the venue
- **title** (`base.String`) – Name of the venue
- **address** (`base.String`) – Address of the venue
- **foursquare\_id** (`typing.Optional[base.String]`) – Foursquare identifier of the venue
- **foursquare\_type** (`typing.Optional[base.String]`) – Foursquare type of the venue, if known
- **google\_place\_id** (`typing.Optional[base.String]`) – Google Places identifier of the venue
- **google\_place\_type** (`typing.Optional[base.String]`) – Google Places type of the venue. See supported types: [https://developers.google.com/places/web-service/supported\\_types](https://developers.google.com/places/web-service/supported_types)
- **disable\_notification** (`typing.Optional[base.Boolean]`) – Sends the message silently. Users will receive a notification with no sound
- **reply\_to\_message\_id** (`typing.Optional[base.Integer]`) – If the message is a reply, ID of the original message
- **allow\_sending\_without\_reply** (`typing.Optional[base.Boolean]`) – Pass `True`, if the message should be sent even if the specified replied-to message is not found

- **reply\_markup** (typing.Union[types.InlineKeyboardMarkup, types.ReplyKeyboardMarkup, types.ReplyKeyboardRemove, types.ForceReply, None]) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user

**Returns** On success, the sent Message is returned

**Return type** types.Message

```
async def send_contact(chat_id: Union[Integer, String], phone_number: String, first_name: String, last_name: Optional[String] = None, vcard: Optional[String] = None, disable_notification: Optional[Boolean] = None, reply_to_message_id: Optional[Integer] = None, allow_sending_without_reply: Optional[Boolean] = None, reply_markup: Optional[Union[aiogram.types.inline_keyboard.InlineKeyboardMarkup, aiogram.types.reply_keyboard.ReplyKeyboardMarkup, aiogram.types.reply_keyboard.ReplyKeyboardRemove, aiogram.types.force_reply.ForceReply]] = None) → aiogram.types.message.Message
```

Use this method to send phone contacts.

Source: <https://core.telegram.org/bots/api#sendcontact>

#### Parameters

- **chat\_id** (typing.Union[base.Integer, base.String]) – Unique identifier for the target chat or username of the target channel
- **phone\_number** (base.String) – Contact's phone number
- **first\_name** (base.String) – Contact's first name
- **last\_name** (typing.Optional[base.String]) – Contact's last name
- **vcard** (typing.Optional[base.String]) – vcard
- **disable\_notification** (typing.Optional[base.Boolean]) – Sends the message silently. Users will receive a notification with no sound
- **reply\_to\_message\_id** (typing.Optional[base.Integer]) – If the message is a reply, ID of the original message
- **allow\_sending\_without\_reply** (typing.Optional[base.Boolean]) – Pass True, if the message should be sent even if the specified replied-to message is not found
- **reply\_markup** (typing.Union[types.InlineKeyboardMarkup, types.ReplyKeyboardMarkup, types.ReplyKeyboardRemove, types.ForceReply, None]) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user

**Returns** On success, the sent Message is returned

**Return type** types.Message



```
async send_poll(chat_id: Union[Integer, String], question: String, options: List[String],
                 is_anonymous: Optional[Boolean] = None, type: Optional[String]
                 = None, allows_multiple_answers: Optional[Boolean] = None, cor-
                 rect_option_id: Optional[Integer] = None, explanation: Optional[String]
                 = None, explanation_parse_mode: Optional[String] = None, explana-
                 tion_entities: Optional[List[aiogram.types.message_entity.MessageEntity]]
                 = None, open_period: Optional[Integer] = None, close_date: Op-
                 tional[Union[Integer, datetime.datetime, datetime.timedelta]] = None,
                 is_closed: Optional[Boolean] = None, disable_notification: Op-
                 tional[Boolean] = None, reply_to_message_id: Optional[Integer] = None,
                 allow_sending_without_reply: Optional[Boolean] = None, reply_markup:
                 Optional[Union[aiogram.types.inline_keyboard.InlineKeyboardMarkup,
                 aiogram.types.reply_keyboard.ReplyKeyboardMarkup,
                 aiogram.types.reply_keyboard.ReplyKeyboardRemove,
                 aiogram.types.force_reply.ForceReply]] = None) →
                 aiogram.types.message.Message
```

Use this method to send a native poll. On success, the sent Message is returned.

Source: <https://core.telegram.org/bots/api#sendpoll>

#### Parameters

- **chat\_id** (typing.Union[base.Integer, base.String]) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- **question** (base.String) – Poll question, 1-300 characters
- **options** (typing.List[base.String]) – A list of answer options, 2-10 strings 1-100 characters each
- **is\_anonymous** (typing.Optional[base.Boolean]) – True, if the poll needs to be anonymous, defaults to True
- **type** (typing.Optional[base.String]) – Poll type, “quiz” or “regular”, defaults to “regular”
- **allows\_multiple\_answers** (typing.Optional[base.Boolean]) – True, if the poll allows multiple answers, ignored for polls in quiz mode, defaults to False
- **correct\_option\_id** (typing.Optional[base.Integer]) – 0-based identifier of the correct answer option, required for polls in quiz mode
- **explanation** (typing.Optional[base.String]) – Text that is shown when a user chooses an incorrect answer or taps on the lamp icon in a quiz-style poll, 0-200 characters with at most 2 line feeds after entities parsing
- **explanation\_parse\_mode** (typing.Optional[base.String]) – Mode for parsing entities in the explanation. See formatting options for more details.
- **explanation\_entities** (typing.Optional[typing.List[types.MessageEntity]]) – List of special entities that appear in message text, which can be specified instead of parse\_mode
- **open\_period** (typing.Optional[base.Integer]) – Amount of time in seconds the poll will be active after creation, 5-600. Can’t be used together with close\_date.
- **close\_date** (typing.Union[base.Integer, datetime.datetime, datetime.timedelta, None]) – Point in time (Unix timestamp) when the poll will be automatically closed. Must be at least 5 and no more than 600 seconds in the future. Can’t be used together with open\_period.



- **is\_closed** (`typing.Optional[base.Boolean]`) – Pass True, if the poll needs to be immediately closed
- **disable\_notification** (`typing.Optional[Boolean]`) – Sends the message silently. Users will receive a notification with no sound.
- **reply\_to\_message\_id** (`typing.Optional[Integer]`) – If the message is a reply, ID of the original message
- **allow\_sending\_without\_reply** (`typing.Optional[base.Boolean]`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- **reply\_markup** (`typing.Union[types.InlineKeyboardMarkup, types.ReplyKeyboardMarkup, types.ReplyKeyboardRemove, types.ForceReply, None]`) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user

**Returns** On success, the sent Message is returned

**Return type** `types.Message`

```
async send_dice(chat_id: Union[Integer, String], disable_notification: Optional[Boolean] = None,
               emoji: Optional[String] = None, reply_to_message_id: Optional[Integer] =
               None, allow_sending_without_reply: Optional[Boolean] = None, reply_markup:
               Optional[Union[aiogram.types.inline_keyboard.InlineKeyboardMarkup,
               aiogram.types.reply_keyboard.ReplyKeyboardMarkup,
               aiogram.types.reply_keyboard.ReplyKeyboardRemove,
               aiogram.types.force_reply.ForceReply]] = None) →
               aiogram.types.message.Message
```

Use this method to send an animated emoji that will display a random value. On success, the sent Message is returned.

Source: <https://core.telegram.org/bots/api#senddice>

#### Parameters

- **chat\_id** (`typing.Union[base.Integer, base.String]`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- **emoji** (`typing.Optional[base.String]`) – Emoji on which the dice throw animation is based. Currently, must be one of “”, “”, “”, “”, or “”. Dice can have values 1-6 for “” and “”, values 1-5 for “” and “”, and values 1-64 for “”. Defaults to “”
- **disable\_notification** (`typing.Optional[base.Boolean]`) – Sends the message silently. Users will receive a notification with no sound
- **reply\_to\_message\_id** (`typing.Optional[base.Integer]`) – If the message is a reply, ID of the original message
- **allow\_sending\_without\_reply** (`typing.Optional[base.Boolean]`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- **reply\_markup** (`typing.Union[types.InlineKeyboardMarkup, types.ReplyKeyboardMarkup, types.ReplyKeyboardRemove, types.ForceReply, None]`) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user

**Returns** On success, the sent Message is returned

**Return type** `types.Message`

**async send\_chat\_action** (*chat\_id: Union[Integer, String], action: String*) → Boolean

Use this method when you need to tell the user that something is happening on the bot's side. The status is set for 5 seconds or less (when a message arrives from your bot, Telegram clients clear its typing status).

We only recommend using this method when a response from the bot will take a noticeable amount of time to arrive.

Source: <https://core.telegram.org/bots/api#sendchataction>

**Parameters**

- **chat\_id** (`typing.Union[base.Integer, base.String]`) – Unique identifier for the target chat or username of the target channel
- **action** (`base.String`) – Type of action to broadcast

**Returns** Returns True on success

**Return type** `base.Boolean`

**async get\_user\_profile\_photos** (*user\_id: Integer, offset: Optional[Integer] = None, limit: Optional[Integer] = None*) → *aiogram.types.user\_profile\_photos.UserProfilePhotos*

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

Source: <https://core.telegram.org/bots/api#getuserprofilephotos>

**Parameters**

- **user\_id** (`base.Integer`) – Unique identifier of the target user
- **offset** (`typing.Optional[base.Integer]`) – Sequential number of the first photo to be returned. By default, all photos are returned
- **limit** (`typing.Optional[base.Integer]`) – Limits the number of photos to be retrieved. Values between 1—100 are accepted. Defaults to 100

**Returns** Returns a UserProfilePhotos object

**Return type** `types.UserProfilePhotos`

**async get\_file** (*file\_id: String*) → *aiogram.types.file.File*

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

Note: This function may not preserve the original file name and MIME type. You should save the file's MIME type and name (if available) when the File object is received.

Source: <https://core.telegram.org/bots/api#getfile>

**Parameters** **file\_id** (`base.String`) – File identifier to get info about

**Returns** On success, a File object is returned

**Return type** `types.File`

**async kick\_chat\_member** (*chat\_id: Union[Integer, String], user\_id: Integer, until\_date: Optional[Union[Integer, datetime.datetime, datetime.timedelta]] = None*) → Boolean

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

The bot must be an administrator in the chat for this to work and must have the appropriate admin rights.

Note: In regular groups (non-supergroups), this method will only work if the ‘All Members Are Admins’ setting is off in the target group. Otherwise members may only be removed by the group’s creator or by the member that added them.

Source: <https://core.telegram.org/bots/api#kickchatmember>

#### Parameters

- **chat\_id** (`typing.Union[base.Integer, base.String]`) – Unique identifier for the target group or username of the target supergroup or channel
- **user\_id** (`base.Integer`) – Unique identifier of the target user
- **until\_date** (`typing.Optional[base.Integer]`) – Date when the user will be unbanned, unix time

**Returns** Returns True on success

**Return type** `base.Boolean`

**async unban\_chat\_member** (*chat\_id: Union[Integer, String], user\_id: Integer, only\_if\_banned: Optional[Boolean] = None*) → Boolean

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

Source: <https://core.telegram.org/bots/api#unbanchatmember>

#### Parameters

- **chat\_id** (`typing.Union[base.Integer, base.String]`) – Unique identifier for the target group or username of the target supergroup or channel (in the format `@username`)
- **user\_id** (`base.Integer`) – Unique identifier of the target user
- **only\_if\_banned** (`typing.Optional[base.Boolean]`) – Do nothing if the user is not banned

**Returns** Returns True on success

**Return type** `base.Boolean`

**async restrict\_chat\_member** (*chat\_id: Union[Integer, String], user\_id: Integer, permissions: Optional[aiogram.types.chat\_permissions.ChatPermissions] = None, until\_date: Optional[Union[Integer, datetime.datetime, datetime.timedelta]] = None, can\_send\_messages: Optional[Boolean] = None, can\_send\_media\_messages: Optional[Boolean] = None, can\_send\_other\_messages: Optional[Boolean] = None, can\_add\_web\_page\_previews: Optional[Boolean] = None*) → Boolean

Use this method to restrict a user in a supergroup. The bot must be an administrator in the supergroup for this to work and must have the appropriate admin rights. Pass True for all boolean parameters to lift restrictions from a user.

Source: <https://core.telegram.org/bots/api#restrictchatmember>

#### Parameters

- **chat\_id** (`typing.Union[base.Integer, base.String]`) – Unique identifier for the target chat or username of the target supergroup

- **user\_id** (`base.Integer`) – Unique identifier of the target user
- **permissions** (`ChatPermissions`) – New user permissions
- **until\_date** (`typing.Optional[base.Integer]`) – Date when restrictions will be lifted for the user, unix time
- **can\_send\_messages** (`typing.Optional[base.Boolean]`) – Pass True, if the user can send text messages, contacts, locations and venues
- **can\_send\_media\_messages** (`typing.Optional[base.Boolean]`) – Pass True, if the user can send audios, documents, photos, videos, video notes and voice notes, implies `can_send_messages`
- **can\_send\_other\_messages** (`typing.Optional[base.Boolean]`) – Pass True, if the user can send animations, games, stickers and use inline bots, implies `can_send_media_messages`
- **can\_add\_web\_page\_previews** (`typing.Optional[base.Boolean]`) – Pass True, if the user may add web page previews to their messages, implies `can_send_media_messages`

**Returns** Returns True on success

**Return type** `base.Boolean`

```
async promote_chat_member (chat_id: Union[Integer, String], user_id: Integer,  
                             is_anonymous: Optional[Boolean] = None, can_change_info:  
                             Optional[Boolean] = None, can_post_messages: Op-  
                             tional[Boolean] = None, can_edit_messages: Op-  
                             tional[Boolean] = None, can_delete_messages: Op-  
                             tional[Boolean] = None, can_invite_users: Optional[Boolean]  
                             = None, can_restrict_members: Optional[Boolean] =  
                             None, can_pin_messages: Optional[Boolean] = None,  
                             can_promote_members: Optional[Boolean] = None) → Boolean
```

Use this method to promote or demote a user in a supergroup or a channel. The bot must be an administrator in the chat for this to work and must have the appropriate admin rights. Pass False for all boolean parameters to demote a user.

Source: <https://core.telegram.org/bots/api#promotechatmember>

#### Parameters

- **chat\_id** (`typing.Union[base.Integer, base.String]`) – Unique identifier for the target chat or username of the target channel
- **user\_id** (`base.Integer`) – Unique identifier of the target user
- **is\_anonymous** (`typing.Optional[base.Boolean]`) – Pass True, if the administrator's presence in the chat is hidden
- **can\_change\_info** (`typing.Optional[base.Boolean]`) – Pass True, if the administrator can change chat title, photo and other settings
- **can\_post\_messages** (`typing.Optional[base.Boolean]`) – Pass True, if the administrator can create channel posts, channels only
- **can\_edit\_messages** (`typing.Optional[base.Boolean]`) – Pass True, if the administrator can edit messages of other users, channels only
- **can\_delete\_messages** (`typing.Optional[base.Boolean]`) – Pass True, if the administrator can delete messages of other users

- **can\_invite\_users** (`typing.Optional[base.Boolean]`) – Pass True, if the administrator can invite new users to the chat
- **can\_restrict\_members** (`typing.Optional[base.Boolean]`) – Pass True, if the administrator can restrict, ban or unban chat members
- **can\_pin\_messages** (`typing.Optional[base.Boolean]`) – Pass True, if the administrator can pin messages, supergroups only
- **can\_promote\_members** (`typing.Optional[base.Boolean]`) – Pass True, if the administrator can add new administrators with a subset of his own privileges or demote administrators that he has promoted, directly or indirectly (promoted by administrators that were appointed by him)

**Returns** Returns True on success

**Return type** `base.Boolean`

**async set\_chat\_administrator\_custom\_title** (*chat\_id: Union[Integer, String], user\_id: Integer, custom\_title: String*) → Boolean

Use this method to set a custom title for an administrator in a supergroup promoted by the bot.

Returns True on success.

Source: <https://core.telegram.org/bots/api#setchatadministratorcustomtitle>

#### Parameters

- **chat\_id** – Unique identifier for the target chat or username of the target supergroup
- **user\_id** – Unique identifier of the target user
- **custom\_title** – New custom title for the administrator; 0-16 characters, emoji are not allowed

**Returns** True on success.

**async set\_chat\_permissions** (*chat\_id: Union[Integer, String], permissions: aiogram.types.chat\_permissions.ChatPermissions*) → Boolean

Use this method to set default chat permissions for all members. The bot must be an administrator in the group or a supergroup for this to work and must have the `can_restrict_members` admin rights.

Returns True on success.

#### Parameters

- **chat\_id** – Unique identifier for the target chat or username of the target supergroup
- **permissions** – New default chat permissions

**Returns** True on success.

**async export\_chat\_invite\_link** (*chat\_id: Union[Integer, String]*) → String

Use this method to generate a new invite link for a chat; any previously generated link is revoked. The bot must be an administrator in the chat for this to work and must have the appropriate admin rights.

Source: <https://core.telegram.org/bots/api#exportchatinvitelink>

**Parameters** **chat\_id** (`typing.Union[base.Integer, base.String]`) – Unique identifier for the target chat or username of the target channel

**Returns** Returns exported invite link as String on success

**Return type** `base.String`

**async set\_chat\_photo** (*chat\_id: Union[Integer, String], photo: InputFile*) → Boolean

Use this method to set a new profile photo for the chat. Photos can't be changed for private chats. The bot must be an administrator in the chat for this to work and must have the appropriate admin rights.

Note: In regular groups (non-supergroups), this method will only work if the 'All Members Are Admins' setting is off in the target group.

Source: <https://core.telegram.org/bots/api#setchatphoto>

**Parameters**

- **chat\_id** (*typing.Union[base.Integer, base.String]*) – Unique identifier for the target chat or username of the target channel
- **photo** (*base.InputFile*) – New chat photo, uploaded using multipart/form-data

**Returns** Returns True on success

**Return type** *base.Boolean*

**async delete\_chat\_photo** (*chat\_id: Union[Integer, String]*) → Boolean

Use this method to delete a chat photo. Photos can't be changed for private chats. The bot must be an administrator in the chat for this to work and must have the appropriate admin rights.

Note: In regular groups (non-supergroups), this method will only work if the 'All Members Are Admins' setting is off in the target group.

Source: <https://core.telegram.org/bots/api#deletechatphoto>

**Parameters** **chat\_id** (*typing.Union[base.Integer, base.String]*) – Unique identifier for the target chat or username of the target channel

**Returns** Returns True on success

**Return type** *base.Boolean*

**async set\_chat\_title** (*chat\_id: Union[Integer, String], title: String*) → Boolean

Use this method to change the title of a chat. Titles can't be changed for private chats. The bot must be an administrator in the chat for this to work and must have the appropriate admin rights.

Note: In regular groups (non-supergroups), this method will only work if the 'All Members Are Admins' setting is off in the target group.

Source: <https://core.telegram.org/bots/api#setchattitle>

**Parameters**

- **chat\_id** (*typing.Union[base.Integer, base.String]*) – Unique identifier for the target chat or username of the target channel
- **title** (*base.String*) – New chat title, 1-255 characters

**Returns** Returns True on success

**Return type** *base.Boolean*

**async set\_chat\_description** (*chat\_id: Union[Integer, String], description: Optional[String] = None*) → Boolean

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

Source: <https://core.telegram.org/bots/api#setchatdescription>

**Parameters**

- **chat\_id** (*typing.Union[base.Integer, base.String]*) – Unique identifier for the target chat or username of the target channel

- **description** (`typing.Optional[base.String]`) – New chat description, 0-255 characters

**Returns** Returns True on success

**Return type** `base.Boolean`

**async pin\_chat\_message** (`chat_id: Union[Integer, String]`, `message_id: Integer`, `disable_notification: Optional[Boolean] = None`) → `Boolean`

Use this method to add a message to the list of pinned messages in a chat. If the chat is not a private chat, the bot must be an administrator in the chat for this to work and must have the ‘can\_pin\_messages’ admin right in a supergroup or ‘can\_edit\_messages’ admin right in a channel. Returns True on success.

Source: <https://core.telegram.org/bots/api#pinchatmessage>

#### Parameters

- **chat\_id** (`typing.Union[base.Integer, base.String]`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- **message\_id** (`base.Integer`) – Identifier of a message to pin
- **disable\_notification** (`typing.Optional[base.Boolean]`) – Pass True, if it is not necessary to send a notification to all group members about the new pinned message

**Returns** Returns True on success

**Return type** `base.Boolean`

**async unpin\_chat\_message** (`chat_id: Union[Integer, String]`, `message_id: Optional[Integer] = None`) → `Boolean`

Use this method to remove a message from the list of pinned messages in a chat. If the chat is not a private chat, the bot must be an administrator in the chat for this to work and must have the ‘can\_pin\_messages’ admin right in a supergroup or ‘can\_edit\_messages’ admin right in a channel. Returns True on success.

Source: <https://core.telegram.org/bots/api#unpinchatmessage>

#### Parameters

- **chat\_id** (`typing.Union[base.Integer, base.String]`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- **message\_id** (`typing.Optional[base.Integer]`) – Identifier of a message to unpin. If not specified, the most recent pinned message (by sending date) will be unpinned.

**Returns** Returns True on success

**Return type** `base.Boolean`

**async unpin\_all\_chat\_messages** (`chat_id: Union[Integer, String]`) → `Boolean`

Use this method to clear the list of pinned messages in a chat. If the chat is not a private chat, the bot must be an administrator in the chat for this to work and must have the ‘can\_pin\_messages’ admin right in a supergroup or ‘can\_edit\_messages’ admin right in a channel. Returns True on success.

Source: <https://core.telegram.org/bots/api#unpinallchatmessages>

**Parameters** **chat\_id** (`typing.Union[base.Integer, base.String]`) – Unique identifier for the target chat or username of the target channel (in the format @channelusername)

**Returns** Returns True on success

**Return type** `base.Boolean`



**async leave\_chat** (*chat\_id: Union[Integer, String]*) → Boolean

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

Source: <https://core.telegram.org/bots/api#leavechat>

**Parameters** **chat\_id** (*typing.Union[base.Integer, base.String]*) – Unique identifier for the target chat or username of the target supergroup or channel

**Returns** Returns True on success

**Return type** *base.Boolean*

**async get\_chat** (*chat\_id: Union[Integer, String]*) → *aiogram.types.chat.Chat*

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

Source: <https://core.telegram.org/bots/api#getchat>

**Parameters** **chat\_id** (*typing.Union[base.Integer, base.String]*) – Unique identifier for the target chat or username of the target supergroup or channel

**Returns** Returns a Chat object on success

**Return type** *types.Chat*

**async get\_chat\_administrators** (*chat\_id: Union[Integer, String]*) → *List[aiogram.types.chat\_member.ChatMember]*

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

Source: <https://core.telegram.org/bots/api#getchatadministrators>

**Parameters** **chat\_id** (*typing.Union[base.Integer, base.String]*) – Unique identifier for the target chat or username of the target supergroup or channel

**Returns** On success, returns an Array of ChatMember objects that contains information about all chat administrators except other bots. If the chat is a group or a supergroup and no administrators were appointed, only the creator will be returned.

**Return type** *typing.List[types.ChatMember]*

**async get\_chat\_members\_count** (*chat\_id: Union[Integer, String]*) → Integer

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

Source: <https://core.telegram.org/bots/api#getchatmemberscount>

**Parameters** **chat\_id** (*typing.Union[base.Integer, base.String]*) – Unique identifier for the target chat or username of the target supergroup or channel

**Returns** Returns Int on success

**Return type** *base.Integer*

**async get\_chat\_member** (*chat\_id: Union[Integer, String], user\_id: Integer*) → *aiogram.types.chat\_member.ChatMember*

Use this method to get information about a member of a chat.

Source: <https://core.telegram.org/bots/api#getchatmember>

**Parameters**

- **chat\_id** (*typing.Union[base.Integer, base.String]*) – Unique identifier for the target chat or username of the target supergroup or channel
- **user\_id** (*base.Integer*) – Unique identifier of the target user

**Returns** Returns a ChatMember object on success



**Return type** `types.ChatMember`

**async set\_chat\_sticker\_set** (*chat\_id*: `Union[Integer, String]`, *sticker\_set\_name*: `String`) → `Boolean`

Use this method to set a new group sticker set for a supergroup. The bot must be an administrator in the chat for this to work and must have the appropriate admin rights.

Use the field `can_set_sticker_set` optionally returned in `getChat` requests to check if the bot can use this method.

Source: <https://core.telegram.org/bots/api#setchatstickerset>

#### Parameters

- **chat\_id** (`typing.Union[base.Integer, base.String]`) – Unique identifier for the target chat or username of the target supergroup
- **sticker\_set\_name** (`base.String`) – Name of the sticker set to be set as the group sticker set

**Returns** Returns True on success

**Return type** `base.Boolean`

**async delete\_chat\_sticker\_set** (*chat\_id*: `Union[Integer, String]`) → `Boolean`

Use this method to delete a group sticker set from a supergroup. The bot must be an administrator in the chat for this to work and must have the appropriate admin rights.

Use the field `can_set_sticker_set` optionally returned in `getChat` requests to check if the bot can use this method.

Source: <https://core.telegram.org/bots/api#deletechatstickerset>

**Parameters** **chat\_id** (`typing.Union[base.Integer, base.String]`) – Unique identifier for the target chat or username of the target supergroup

**Returns** Returns True on success

**Return type** `base.Boolean`

**async answer\_callback\_query** (*callback\_query\_id*: `String`, *text*: `Optional[String] = None`, *show\_alert*: `Optional[Boolean] = None`, *url*: `Optional[String] = None`, *cache\_time*: `Optional[Integer] = None`) → `Boolean`

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

Alternatively, the user can be redirected to the specified Game URL. For this option to work, you must first create a game for your bot via @Botfather and accept the terms. Otherwise, you may use links like `t.me/your_bot?start=XXXX` that open your bot with a parameter.

Source: <https://core.telegram.org/bots/api#answercallbackquery>

#### Parameters

- **callback\_query\_id** (`base.String`) – Unique identifier for the query to be answered
- **text** (`typing.Optional[base.String]`) – Text of the notification. If not specified, nothing will be shown to the user, 0-1024 characters
- **show\_alert** (`typing.Optional[base.Boolean]`) – If true, an alert will be shown by the client instead of a notification at the top of the chat screen. Defaults to false.

- **url** (`typing.Optional[base.String]`) – URL that will be opened by the user’s client
- **cache\_time** (`typing.Optional[base.Integer]`) – The maximum amount of time in seconds that the result of the callback query may be cached client-side.

**Returns** On success, True is returned

**Return type** `base.Boolean`

**async set\_my\_commands** (*commands:* `List[aiogram.types.bot_command.BotCommand]`) → `Boolean`

Use this method to change the list of the bot’s commands.

Source: <https://core.telegram.org/bots/api#setmycommands>

**Parameters** **commands** – A JSON-serialized list of bot commands to be set as the list of the bot’s commands. At most 100 commands can be specified.

**Returns** Returns True on success.

**Return type** `base.Boolean`

**async get\_my\_commands** () → `List[aiogram.types.bot_command.BotCommand]`

Use this method to get the current list of the bot’s commands.

Source: <https://core.telegram.org/bots/api#getmycommands> :return: Returns Array of BotCommand on success. :rtype: `typing.List[types.BotCommand]`

**async edit\_message\_text** (*text:* `String`, *chat\_id:* `Optional[Union[Integer, String]] = None`, *message\_id:* `Optional[Integer] = None`, *inline\_message\_id:* `Optional[String] = None`, *parse\_mode:* `Optional[String] = None`, *entities:* `Optional[List[aiogram.types.message_entity.MessageEntity]] = None`, *disable\_web\_page\_preview:* `Optional[Boolean] = None`, *reply\_markup:* `Optional[aiogram.types.inline_keyboard.InlineKeyboardMarkup] = None`) → `aiogram.types.message.Message`

Use this method to edit text and game messages sent by the bot or via the bot (for inline bots).

Source: <https://core.telegram.org/bots/api#editmessagetext>

#### Parameters

- **chat\_id** (`typing.Union[base.Integer, base.String, None]`) – Required if `inline_message_id` is not specified Unique identifier for the target chat or username of the target channel
- **message\_id** (`typing.Optional[base.Integer]`) – Required if `inline_message_id` is not specified. Identifier of the sent message
- **inline\_message\_id** (`typing.Optional[base.String]`) – Required if `chat_id` and `message_id` are not specified. Identifier of the inline message
- **text** (`base.String`) – New text of the message
- **parse\_mode** (`typing.Optional[base.String]`) – Send Markdown or HTML, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in your bot’s message.
- **entities** (`typing.Optional[typing.List[types.MessageEntity]]`) – List of special entities that appear in message text, which can be specified instead of `parse_mode`

- **disable\_web\_page\_preview** (`typing.Optional[base.Boolean]`) – Disables link previews for links in this message
- **reply\_markup** (`typing.Optional[types.InlineKeyboardMarkup]`) – A JSON-serialized object for an inline keyboard

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

**Return type** `typing.Union[types.Message, base.Boolean]`

```
async edit_message_caption (chat_id: Optional[Union[Integer, String]] = None, message_id: Optional[Integer] = None, inline_message_id: Optional[String] = None, caption: Optional[String] = None, parse_mode: Optional[String] = None, caption_entities: Optional[List[aiogram.types.message_entity.MessageEntity]] = None, reply_markup: Optional[aiogram.types.inline_keyboard.InlineKeyboardMarkup] = None) → aiogram.types.message.Message
```

Use this method to edit captions of messages sent by the bot or via the bot (for inline bots).

Source: <https://core.telegram.org/bots/api#editmessagecaption>

#### Parameters

- **chat\_id** (`typing.Union[base.Integer, base.String, None]`) – Required if `inline_message_id` is not specified Unique identifier for the target chat or username of the target channel
- **message\_id** (`typing.Optional[base.Integer]`) – Required if `inline_message_id` is not specified. Identifier of the sent message
- **inline\_message\_id** (`typing.Optional[base.String]`) – Required if `chat_id` and `message_id` are not specified. Identifier of the inline message
- **caption** (`typing.Optional[base.String]`) – New caption of the message
- **parse\_mode** (`typing.Optional[base.String]`) – Send Markdown or HTML, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in your bot's message.
- **caption\_entities** (`typing.Optional[typing.List[types.MessageEntity]]`) – List of special entities that appear in message text, which can be specified instead of `parse_mode`
- **reply\_markup** (`typing.Optional[types.InlineKeyboardMarkup]`) – A JSON-serialized object for an inline keyboard

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

**Return type** `typing.Union[types.Message, base.Boolean]`

```
async edit_message_media (media: aiogram.types.input_media.InputMedia, chat_id: Optional[Union[Integer, String]] = None, message_id: Optional[Integer] = None, inline_message_id: Optional[String] = None, reply_markup: Optional[aiogram.types.inline_keyboard.InlineKeyboardMarkup] = None) → Union[aiogram.types.message.Message, Boolean]
```

Use this method to edit audio, document, photo, or video messages. If a message is a part of a message album, then it can be edited only to a photo or a video. Otherwise, message type can be changed arbitrarily.

When inline message is edited, new file can't be uploaded. Use previously uploaded file via its `file_id` or specify a URL.

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

Source <https://core.telegram.org/bots/api#editmessagemedia>

#### Parameters

- **chat\_id** (`typing.Union[typing.Union[base.Integer, base.String], None]`) – Required if `inline_message_id` is not specified
- **message\_id** (`typing.Optional[base.Integer]`) – Required if `inline_message_id` is not specified. Identifier of the sent message
- **inline\_message\_id** (`typing.Optional[base.String]`) – Required if `chat_id` and `message_id` are not specified. Identifier of the inline message
- **media** (`types.InputMedia`) – A JSON-serialized object for a new media content of the message
- **reply\_markup** (`typing.Optional[types.InlineKeyboardMarkup]`) – A JSON-serialized object for a new inline keyboard

**Returns** On success, if the edited message was sent by the bot, the edited Message is returned, otherwise True is returned

**Return type** `typing.Union[types.Message, base.Boolean]`

```
async edit_message_reply_markup (chat_id: Optional[Union[Integer, String]] = None, message_id: Optional[Integer] = None, inline_message_id: Optional[String] = None, reply_markup: Optional[aiogram.types.inline_keyboard.InlineKeyboardMarkup] = None) → aiogram.types.message.Message
```

Use this method to edit only the reply markup of messages sent by the bot or via the bot (for inline bots).

Source: <https://core.telegram.org/bots/api#editmessagereplymarkup>

#### Parameters

- **chat\_id** (`typing.Union[base.Integer, base.String, None]`) – Required if `inline_message_id` is not specified Unique identifier for the target chat or username of the target channel
- **message\_id** (`typing.Optional[base.Integer]`) – Required if `inline_message_id` is not specified. Identifier of the sent message
- **inline\_message\_id** (`typing.Optional[base.String]`) – Required if `chat_id` and `message_id` are not specified. Identifier of the inline message
- **reply\_markup** (`typing.Optional[types.InlineKeyboardMarkup]`) – A JSON-serialized object for an inline keyboard

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

**Return type** `typing.Union[types.Message, base.Boolean]`

```
async stop_poll (chat_id: Union[String, Integer], message_id: Integer, reply_markup: Optional[aiogram.types.inline_keyboard.InlineKeyboardMarkup] = None) → aiogram.types.poll.Poll
```

Use this method to stop a poll which was sent by the bot. On success, the stopped Poll with the final results is returned.

**Parameters**

- **chat\_id** (`typing.Union[base.String, base.Integer]`) – Unique identifier for the target chat or username of the target channel
- **message\_id** (`base.Integer`) – Identifier of the original message with the poll
- **reply\_markup** (`typing.Optional[types.InlineKeyboardMarkup]`) – A JSON-serialized object for a new message inline keyboard.

**Returns** On success, the stopped Poll with the final results is returned.

**Return type** `types.Poll`

**async delete\_message** (*chat\_id: Union[Integer, String], message\_id: Integer*) → Boolean

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

Source: <https://core.telegram.org/bots/api#deletemessage>

**Parameters**

- **chat\_id** (`typing.Union[base.Integer, base.String]`) – Unique identifier for the target chat or username of the target channel
- **message\_id** (`base.Integer`) – Identifier of the message to delete

**Returns** Returns True on success

**Return type** `base.Boolean`

**async send\_sticker** (*chat\_id: Union[Integer, String], sticker: Union[InputFile, String], disable\_notification: Optional[Boolean] = None, reply\_to\_message\_id: Optional[Integer] = None, allow\_sending\_without\_reply: Optional[Boolean] = None, reply\_markup: Optional[Union[aiogram.types.inline\_keyboard.InlineKeyboardMarkup, aiogram.types.reply\_keyboard.ReplyKeyboardMarkup, aiogram.types.reply\_keyboard.ReplyKeyboardRemove, aiogram.types.force\_reply.ForceReply]] = None*) → *aiogram.types.message.Message*

Use this method to send .webp stickers.

Source: <https://core.telegram.org/bots/api#sendsticker>

**Parameters**

- **chat\_id** (`typing.Union[base.Integer, base.String]`) – Unique identifier for the target chat or username of the target channel
- **sticker** (`typing.Union[base.InputFile, base.String]`) – Sticker to send
- **disable\_notification** (`typing.Optional[base.Boolean]`) – Sends the message silently. Users will receive a notification with no sound
- **reply\_to\_message\_id** (`typing.Optional[base.Integer]`) – If the message is a reply, ID of the original message

- **allow\_sending\_without\_reply** (`typing.Optional[base.Boolean]`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- **reply\_markup** (`typing.Union[types.InlineKeyboardMarkup, types.ReplyKeyboardMarkup, types.ReplyKeyboardRemove, types.ForceReply, None]`) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user

**Returns** On success, the sent Message is returned

**Return type** `types.Message`

**async get\_sticker\_set** (*name: String*) → *aiogram.types.sticker\_set.StickerSet*

Use this method to get a sticker set.

Source: <https://core.telegram.org/bots/api#getstickerset>

**Parameters** **name** (`base.String`) – Name of the sticker set

**Returns** On success, a StickerSet object is returned

**Return type** `types.StickerSet`

**async upload\_sticker\_file** (*user\_id: Integer, png\_sticker: InputFile*) → *aiogram.types.file.File*

Use this method to upload a .png file with a sticker for later use in `createNewStickerSet` and `addStickerToSet` methods (can be used multiple times).

Source: <https://core.telegram.org/bots/api#uploadstickerfile>

**Parameters**

- **user\_id** (`base.Integer`) – User identifier of sticker file owner
- **png\_sticker** (`base.InputFile`) – 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.

**Returns** Returns the uploaded File on success

**Return type** `types.File`

**async create\_new\_sticker\_set** (*user\_id: Integer, name: String, title: String, emojis: String, png\_sticker: Optional[Union[InputFile, String]] = None, tgs\_sticker: Optional[InputFile] = None, contains\_masks: Optional[Boolean] = None, mask\_position: Optional[aiogram.types.mask\_position.MaskPosition] = None*) → Boolean

Use this method to create a new sticker set owned by a user. The bot will be able to edit the sticker set thus created. You must use exactly one of the fields `png_sticker` or `tgs_sticker`.

Source: <https://core.telegram.org/bots/api#createnewstickerset>

**Parameters**

- **user\_id** (`base.Integer`) – User identifier of created sticker set owner
- **name** (`base.String`) – Short name of sticker set, to be used in `t.me/addstickers/` URLs (e.g., animals). Can contain only english letters, digits and underscores. Must begin with a letter, can't contain consecutive underscores and must end in “\_by\_<bot username>”. <bot\_username> is case insensitive. 1-64 characters.
- **title** (`base.String`) – Sticker set title, 1-64 characters

- **png\_sticker** (`typing.Union[base.InputFile, base.String]`) – PNG image with the sticker, must be up to 512 kilobytes in size, dimensions must not exceed 512px, and either width or height must be exactly 512px. Pass a `file_id` as a `String` to send a file that already exists on the Telegram servers, pass an HTTP URL as a `String` for Telegram to get a file from the Internet, or upload a new one using multipart/form-data. More info on <https://core.telegram.org/bots/api#sending-files>
- **tgs\_sticker** (`base.InputFile`) – TGS animation with the sticker, uploaded using multipart/form-data. See [https://core.telegram.org/animated\\_stickers#technical-requirements](https://core.telegram.org/animated_stickers#technical-requirements) for technical requirements
- **emojis** (`base.String`) – One or more emoji corresponding to the sticker
- **contains\_masks** (`typing.Optional[base.Boolean]`) – Pass `True`, if a set of mask stickers should be created
- **mask\_position** (`typing.Optional[types.MaskPosition]`) – A JSON-serialized object for position where the mask should be placed on faces

**Returns** Returns `True` on success

**Return type** `base.Boolean`

```
async add_sticker_to_set (user_id: Integer, name: String, emojis: String, png_sticker:
                           Optional[Union[InputFile, String]] = None, tgs_sticker:
                           Optional[InputFile] = None, mask_position: Op-
                           tional[aiogram.types.mask_position.MaskPosition] = None)
                           → Boolean
```

Use this method to add a new sticker to a set created by the bot. You must use exactly one of the fields `png_sticker` or `tgs_sticker`. Animated stickers can be added to animated sticker sets and only to them. Animated sticker sets can have up to 50 stickers. Static sticker sets can have up to 120 stickers.

Source: <https://core.telegram.org/bots/api#addstickertoset>

#### Parameters

- **user\_id** (`base.Integer`) – User identifier of sticker set owner
- **name** (`base.String`) – Sticker set name
- **png\_sticker** (`typing.Union[base.InputFile, base.String]`) – PNG image with the sticker, must be up to 512 kilobytes in size, dimensions must not exceed 512px, and either width or height must be exactly 512px. Pass a `file_id` as a `String` to send a file that already exists on the Telegram servers, pass an HTTP URL as a `String` for Telegram to get a file from the Internet, or upload a new one using multipart/form-data. More info on <https://core.telegram.org/bots/api#sending-files>
- **tgs\_sticker** (`base.InputFile`) – TGS animation with the sticker, uploaded using multipart/form-data. See [https://core.telegram.org/animated\\_stickers#technical-requirements](https://core.telegram.org/animated_stickers#technical-requirements) for technical requirements
- **emojis** (`base.String`) – One or more emoji corresponding to the sticker
- **mask\_position** (`typing.Optional[types.MaskPosition]`) – A JSON-serialized object for position where the mask should be placed on faces

**Returns** Returns `True` on success

**Return type** `base.Boolean`

```
async set_sticker_position_in_set (sticker: String, position: Integer) → Boolean
```

Use this method to move a sticker in a set created by the bot to a specific position.

Source: <https://core.telegram.org/bots/api#setstickerpositioninset>



**Parameters**

- **sticker** (`base.String`) – File identifier of the sticker
- **position** (`base.Integer`) – New sticker position in the set, zero-based

**Returns** Returns True on success

**Return type** `base.Boolean`

**async delete\_sticker\_from\_set** (*sticker: String*) → Boolean

Use this method to delete a sticker from a set created by the bot.

Source: <https://core.telegram.org/bots/api#deletestickerfromset>

**Parameters** **sticker** (`base.String`) – File identifier of the sticker

**Returns** Returns True on success

**Return type** `base.Boolean`

**async set\_sticker\_set\_thumb** (*name: String, user\_id: Integer, thumb: Optional[Union[InputFile, String]] = None*) → Boolean

Use this method to set the thumbnail of a sticker set. Animated thumbnails can be set for animated sticker sets only.

Source: <https://core.telegram.org/bots/api#setstickersetthumb>

**Parameters**

- **name** (`base.String`) – Sticker set name
- **user\_id** (`base.Integer`) – User identifier of the sticker set owner
- **thumb** (`typing.Union[base.InputFile, base.String]`) – A PNG image with the thumbnail, must be up to 128 kilobytes in size and have width and height exactly 100px, or a TGS animation with the thumbnail up to 32 kilobytes in size; see [https://core.telegram.org/animated\\_stickers#technical-requirements](https://core.telegram.org/animated_stickers#technical-requirements) for animated 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 info on <https://core.telegram.org/bots/api#sending-files>. Animated sticker set thumbnail can't be uploaded via HTTP URL.

**Returns** Returns True on success

**Return type** `base.Boolean`

**async answer\_inline\_query** (*inline\_query\_id: String, results: List[aiogram.types.inline\_query\_result.InlineQueryResult], cache\_time: Optional[Integer] = None, is\_personal: Optional[Boolean] = None, next\_offset: Optional[String] = None, switch\_pm\_text: Optional[String] = None, switch\_pm\_parameter: Optional[String] = None*) → Boolean

Use this method to send answers to an inline query. No more than 50 results per query are allowed.

Source: <https://core.telegram.org/bots/api#answerinlinequery>

**Parameters**

- **inline\_query\_id** (`base.String`) – Unique identifier for the answered query
- **results** (`typing.List[types.InlineQueryResult]`) – A JSON-serialized array of results for the inline query



- **cache\_time** (`typing.Optional[base.Integer]`) – The maximum amount of time in seconds that the result of the inline query may be cached on the server. Defaults to 300.
- **is\_personal** (`typing.Optional[base.Boolean]`) – Pass True, if results may be cached on the server side only for the user that sent the query. By default, results may be returned to any user who sends the same query
- **next\_offset** (`typing.Optional[base.String]`) – Pass the offset that a client should send in the next query with the same text to receive more results. Pass an empty string if there are no more results or if you don't support pagination. Offset length can't exceed 64 bytes.
- **switch\_pm\_text** (`typing.Optional[base.String]`) – If passed, clients will display a button with specified text that switches the user to a private chat with the bot and sends the bot a start message with the parameter `switch_pm_parameter`
- **switch\_pm\_parameter** (`typing.Optional[base.String]`) – Deep-linking parameter for the /start message sent to the bot when user presses the switch button. 1-64 characters, only A-Z, a-z, 0-9, \_ and - are allowed.

**Returns** On success, True is returned

**Return type** `base.Boolean`

```
async send_invoice(chat_id: Integer, title: String, description: String, payload: String,
    provider_token: String, start_parameter: String, currency: String,
    prices: List[aiogram.types.labeled_price.LabeledPrice], provider_data:
    Optional[Dict] = None, photo_url: Optional[String] = None,
    photo_size: Optional[Integer] = None, photo_width: Optional[Integer]
    = None, photo_height: Optional[Integer] = None, need_name: Op-
    tional[Boolean] = None, need_phone_number: Optional[Boolean] =
    None, need_email: Optional[Boolean] = None, need_shipping_address:
    Optional[Boolean] = None, send_phone_number_to_provider: Op-
    tional[Boolean] = None, send_email_to_provider: Optional[Boolean] =
    None, is_flexible: Optional[Boolean] = None, disable_notification: Op-
    tional[Boolean] = None, reply_to_message_id: Optional[Integer] = None,
    allow_sending_without_reply: Optional[Boolean] = None, reply_markup:
    Optional[aiogram.types.inline_keyboard.InlineKeyboardMarkup] = None)
    → aiogram.types.message.Message
```

Use this method to send invoices.

Source: <https://core.telegram.org/bots/api#sendinvoice>

#### Parameters

- **chat\_id** (`base.Integer`) – Unique identifier for the target private chat
- **title** (`base.String`) – Product name, 1-32 characters
- **description** (`base.String`) – Product description, 1-255 characters
- **payload** (`base.String`) – Bot-defined invoice payload, 1-128 bytes This will not be displayed to the user, use for your internal processes.
- **provider\_token** (`base.String`) – Payments provider token, obtained via Botfather
- **start\_parameter** (`base.String`) – Unique deep-linking parameter that can be used to generate this invoice when used as a start parameter

- **currency** (`base.String`) – Three-letter ISO 4217 currency code, see more on currencies
- **prices** (`typing.List[types.LabeledPrice]`) – Price breakdown, a list of components (e.g. product price, tax, discount, delivery cost, delivery tax, bonus, etc.)
- **provider\_data** (`typing.Optional[typing.Dict]`) – JSON-encoded data about the invoice, which will be shared with the payment provider
- **photo\_url** (`typing.Optional[base.String]`) – URL of the product photo for the invoice
- **photo\_size** (`typing.Optional[base.Integer]`) – Photo size
- **photo\_width** (`typing.Optional[base.Integer]`) – Photo width
- **photo\_height** (`typing.Optional[base.Integer]`) – Photo height
- **need\_name** (`typing.Optional[base.Boolean]`) – Pass True, if you require the user's full name to complete the order
- **need\_phone\_number** (`typing.Optional[base.Boolean]`) – Pass True, if you require the user's phone number to complete the order
- **need\_email** (`typing.Optional[base.Boolean]`) – Pass True, if you require the user's email to complete the order
- **need\_shipping\_address** (`typing.Optional[base.Boolean]`) – Pass True, if you require the user's shipping address to complete the order
- **send\_phone\_number\_to\_provider** (`typing.Optional[base.Boolean]`) – Pass True, if user's phone number should be sent to provider
- **send\_email\_to\_provider** (`typing.Optional[base.Boolean]`) – Pass True, if user's email address should be sent to provider
- **is\_flexible** (`typing.Optional[base.Boolean]`) – Pass True, if the final price depends on the shipping method
- **disable\_notification** (`typing.Optional[base.Boolean]`) – Sends the message silently. Users will receive a notification with no sound
- **reply\_to\_message\_id** (`typing.Optional[base.Integer]`) – If the message is a reply, ID of the original message
- **allow\_sending\_without\_reply** (`typing.Optional[base.Boolean]`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- **reply\_markup** (`typing.Optional[types.InlineKeyboardMarkup]`) – A JSON-serialized object for an inline keyboard. If empty, one 'Pay total price' button will be shown. If not empty, the first button must be a Pay button.

**Returns** On success, the sent Message is returned

**Return type** `types.Message`

**async answer\_shipping\_query** (*shipping\_query\_id: String, ok: Boolean, shipping\_options: Optional[List[aiogram.types.shipping\_option.ShippingOption]] = None, error\_message: Optional[String] = None*) → Boolean

If you sent an invoice requesting a shipping address and the parameter `is_flexible` was specified, the Bot API will send an Update with a `shipping_query` field to the bot.

Source: <https://core.telegram.org/bots/api#answershippingquery>

### Parameters

- **shipping\_query\_id** (`base.String`) – Unique identifier for the query to be answered
- **ok** (`base.Boolean`) – Specify True if delivery to the specified address is possible and False if there are any problems (for example, if delivery to the specified address is not possible)
- **shipping\_options** (`typing.Union[typing.List[types.ShippingOption], None]`) – Required if `ok` is True. A JSON-serialized array of available shipping options
- **error\_message** (`typing.Optional[base.String]`) – Required if `ok` is False. Error message in human readable form that explains why it is impossible to complete the order (e.g. “Sorry, delivery to your desired address is unavailable”). Telegram will display this message to the user.

**Returns** On success, True is returned

**Return type** `base.Boolean`

**async answer\_pre\_checkout\_query** (*pre\_checkout\_query\_id: String, ok: Boolean, error\_message: Optional[String] = None*) → `Boolean`

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

Source: <https://core.telegram.org/bots/api#answerprecheckoutquery>

### Parameters

- **pre\_checkout\_query\_id** (`base.String`) – Unique identifier for the query to be answered
- **ok** (`base.Boolean`) – Specify True if everything is alright (goods are available, etc.) and the bot is ready to proceed with the order. Use False if there are any problems.
- **error\_message** (`typing.Optional[base.String]`) – Required if `ok` is False. Error message in human readable form that explains the reason for failure to proceed with the checkout (e.g. “Sorry, somebody just bought the last of our amazing black T-shirts while you were busy filling out your payment details. Please choose a different color or garment!”). Telegram will display this message to the user.

**Returns** On success, True is returned

**Return type** `base.Boolean`

**async set\_passport\_data\_errors** (*user\_id: Integer, errors: List[aiogram.types.passport\_element\_error.PassportElementError]*) → `Boolean`

Informs a user that some of the Telegram Passport elements they provided contains errors. The user will not be able to re-submit their Passport to you until the errors are fixed (the contents of the field for which you returned the error must change). Returns True on success.

Use this if the data submitted by the user doesn’t satisfy the standards your service requires for any reason. For example, if a birthday date seems invalid, a submitted document is blurry, a scan shows evidence of tampering, etc. Supply some details in the error message to make sure the user knows how to correct the issues.

Source: <https://core.telegram.org/bots/api#setpassportdataerrors>

### Parameters

- **user\_id** (`base.Integer`) – User identifier
- **errors** (`typing.List[types.PassportElementError]`) – A JSON-serialized array describing the errors

**Returns** Returns True on success

**Return type** `base.Boolean`

```
async send_game(chat_id: Integer, game_short_name: String, disable_notification: Optional[Boolean] = None, reply_to_message_id: Optional[Integer] = None, allow_sending_without_reply: Optional[Boolean] = None, reply_markup: Optional[aiogram.types.inline_keyboard.InlineKeyboardMarkup] = None) → aiogram.types.message.Message
```

Use this method to send a game.

Source: <https://core.telegram.org/bots/api#sendgame>

#### Parameters

- **chat\_id** (`base.Integer`) – Unique identifier for the target chat
- **game\_short\_name** (`base.String`) – Short name of the game, serves as the unique identifier for the game. Set up your games via Botfather.
- **disable\_notification** (`typing.Optional[base.Boolean]`) – Sends the message silently. Users will receive a notification with no sound
- **reply\_to\_message\_id** (`typing.Optional[base.Integer]`) – If the message is a reply, ID of the original message
- **allow\_sending\_without\_reply** (`typing.Optional[base.Boolean]`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- **reply\_markup** (`typing.Optional[types.InlineKeyboardMarkup]`) – A JSON-serialized object for an inline keyboard If empty, one ‘Play game\_title’ button will be shown. If not empty, the first button must launch the game.

**Returns** On success, the sent Message is returned

**Return type** `types.Message`

```
async set_game_score(user_id: Integer, score: Integer, force: Optional[Boolean] = None, disable_edit_message: Optional[Boolean] = None, chat_id: Optional[Integer] = None, message_id: Optional[Integer] = None, inline_message_id: Optional[String] = None) → aiogram.types.message.Message
```

Use this method to set the score of the specified user in a game.

Source: <https://core.telegram.org/bots/api#setgamescore>

#### Parameters

- **user\_id** (`base.Integer`) – User identifier
- **score** (`base.Integer`) – New score, must be non-negative
- **force** (`typing.Optional[base.Boolean]`) – Pass True, if the high score is allowed to decrease This can be useful when fixing mistakes or banning cheaters
- **disable\_edit\_message** (`typing.Optional[base.Boolean]`) – Pass True, if the game message should not be automatically edited to include the current scoreboard

- **chat\_id** (`typing.Optional[base.Integer]`) – Required if `inline_message_id` is not specified. Unique identifier for the target chat
- **message\_id** (`typing.Optional[base.Integer]`) – Required if `inline_message_id` is not specified. Identifier of the sent message
- **inline\_message\_id** (`typing.Optional[base.String]`) – Required if `chat_id` and `message_id` are not specified. Identifier of the inline message

**Returns** On success, if the message was sent by the bot, returns the edited Message, otherwise returns True. Returns an error, if the new score is not greater than the user's current score in the chat and force is False.

**Return type** `typing.Union[types.Message, base.Boolean]`

```
async get_game_high_scores (user_id: Integer, chat_id: Optional[Integer] =  
None, message_id: Optional[Integer] = None, in-  
line_message_id: Optional[String] = None) →  
List[aiogram.types.game_high_score.GameHighScore]
```

Use this method to get data for high score tables.

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

Source: <https://core.telegram.org/bots/api#getgamehighscores>

#### Parameters

- **user\_id** (`base.Integer`) – Target user id
- **chat\_id** (`typing.Optional[base.Integer]`) – Required if `inline_message_id` is not specified. Unique identifier for the target chat
- **message\_id** (`typing.Optional[base.Integer]`) – Required if `inline_message_id` is not specified. Identifier of the sent message
- **inline\_message\_id** (`typing.Optional[base.String]`) – Required if `chat_id` and `message_id` are not specified. Identifier of the inline message

**Returns** Will return the score of the specified user and several of his neighbors in a game. On success, returns an Array of GameHighScore objects. This method will currently return scores for the target user, plus two of his closest neighbors on each side. Will also return the top three users if the user and his neighbors are not among them.

**Return type** `typing.List[types.GameHighScore]`

## API Helpers

```
class aiogram.bot.api.TelegramAPIServer (base: str, file: str)
```

Bases: `object`

Base config for API Endpoints

```
api_url (token: str, method: str) → str
```

Generate URL for API methods

#### Parameters

- **token** – Bot token
- **method** – API method name (case insensitive)

**Returns** URL

**file\_url** (*token: str, path: str*) → str  
Generate URL for downloading files

**Parameters**

- **token** – Bot token
- **path** – file path

**Returns** URL

**aiogram.bot.api.check\_token** (*token: str*) → bool  
Validate BOT token

**Parameters** **token** –

**Returns**

**aiogram.bot.api.check\_result** (*method\_name: str, content\_type: str, status\_code: int, body: str*)  
Checks whether *result* is a valid API response. A result is considered invalid if: - The server returned an HTTP response code other than 200 - The content of the result is invalid JSON. - The method call was unsuccessful (The JSON 'ok' field equals False)

**Parameters**

- **method\_name** – The name of the method called
- **status\_code** – status code
- **content\_type** – content type of result
- **body** – result body

**Returns** The result parsed to a JSON dictionary

**Raises** **ApiException** – if one of the above listed cases is applicable

**aiogram.bot.api.guess\_filename** (*obj*)  
Get file name from object

**Parameters** **obj** –

**Returns**

**aiogram.bot.api.compose\_data** (*params=None, files=None*)  
Prepare request data

**Parameters**

- **params** –
- **files** –

**Returns**

**class** **aiogram.bot.api.Methods**  
Bases: **aiogram.utils.helper.Helper**  
Helper for Telegram API Methods listed on <https://core.telegram.org/bots/api>  
List is updated to Bot API 5.0

## 4.4.2 Telegram data types

### Bases

#### Base TelegramObject

#### MetaTelegramObject

```
class aiogram.types.base.MetaTelegramObject (name: str, bases: Tuple[Type], namespace: Dict[str, Any], **kwargs: Any)
```

Bases: type

Metaclass for telegram objects

#### TelegramObject

```
class aiogram.types.base.TelegramObject (conf: Optional[Dict[str, Any]] = None, **kwargs: Any)
```

Bases: aiogram.utils.mixins.ContextInstanceMixin

Abstract class for telegram objects

Deserialize object

##### Parameters

- **conf** –
- **kwargs** –

##### property props

Get props

**Returns** dict with props

##### property props\_aliases

Get aliases for props

**Returns**

##### property values

Get values

**Returns**

```
classmethod to_object (data: Dict[str, Any]) → T
```

Deserialize object

**Parameters** **data** –

**Returns**

```
to_python () → Dict[str, Any]
```

Get object as JSON serializable

**Returns**

```
clean () → None
```

Remove empty values

```
as_json () → str
```

Get object as JSON string

**Returns** JSON

**Return type** `str`

**iter\_keys** () → Generator[Any, None, None]  
Iterate over keys

**Returns**

**iter\_values** () → Generator[Any, None, None]  
Iterate over values

**Returns**

## Fields

### BaseField

```
class aiogram.types.fields.BaseField(*, base=None, default=None, alias=None,
                                     on_change=None)
```

Bases: object

Base field (prop)

Init prop

#### Parameters

- **base** – class for child element
- **default** – default value
- **alias** – alias name (for e.g. field ‘from’ has to be named ‘from\_user’ as ‘from’ is a builtin Python keyword)
- **on\_change** – callback will be called when value is changed

**get\_value** (*instance*)  
Get value for the current object instance

**Parameters** **instance** –

**Returns**

**set\_value** (*instance, value, parent=None*)  
Set prop value

#### Parameters

- **instance** –
- **value** –
- **parent** –

**Returns**

**abstract serialize** (*value*)  
Serialize value to python

**Parameters** **value** –

**Returns**



**abstract deserialize** (*value*, *parent=None*)  
 Deserialize python object value to TelegramObject value

**export** (*instance*)  
 Alias for *serialize* but for current Object instance

**Parameters** *instance* –

**Returns**

## Field

**class** aiogram.types.fields.**Field**(\*, *base=None*, *default=None*, *alias=None*,  
*on\_change=None*)

Bases: *aiogram.types.fields.BaseField*

Simple field

Init prop

**Parameters**

- **base** – class for child element
- **default** – default value
- **alias** – alias name (for e.g. field ‘from’ has to be named ‘from\_user’ as ‘from’ is a builtin Python keyword)
- **on\_change** – callback will be called when value is changed

**serialize** (*value*)  
 Serialize value to python

**Parameters** *value* –

**Returns**

**deserialize** (*value*, *parent=None*)  
 Deserialize python object value to TelegramObject value

## ListField

**class** aiogram.types.fields.**ListField**(\*args, \*\*kwargs)

Bases: *aiogram.types.fields.Field*

Field contains list of objects

Init prop

**Parameters**

- **base** – class for child element
- **default** – default value
- **alias** – alias name (for e.g. field ‘from’ has to be named ‘from\_user’ as ‘from’ is a builtin Python keyword)
- **on\_change** – callback will be called when value is changed

**serialize** (*value*)  
 Serialize value to python

**Parameters value –**

**Returns**

**deserialize** (*value*, *parent=None*)

Deserialize python object value to TelegramObject value

## ListOfLists

```
class aiogram.types.fields.ListOfLists(*, base=None, default=None, alias=None,
                                       on_change=None)
```

Bases: *aiogram.types.fields.Field*

Init prop

**Parameters**

- **base** – class for child element
- **default** – default value
- **alias** – alias name (for e.g. field ‘from’ has to be named ‘from\_user’ as ‘from’ is a builtin Python keyword)
- **on\_change** – callback will be called when value is changed

**serialize** (*value*)

Serialize value to python

**Parameters value –**

**Returns**

**deserialize** (*value*, *parent=None*)

Deserialize python object value to TelegramObject value

## DateTimeField

```
class aiogram.types.fields.DateTimeField(*, base=None, default=None, alias=None,
                                           on_change=None)
```

Bases: *aiogram.types.fields.Field*

In this field st\_ored datetime

in: unixtime out: datetime

Init prop

**Parameters**

- **base** – class for child element
- **default** – default value
- **alias** – alias name (for e.g. field ‘from’ has to be named ‘from\_user’ as ‘from’ is a builtin Python keyword)
- **on\_change** – callback will be called when value is changed

**serialize** (*value: datetime.datetime*)

Serialize value to python

**Parameters value –**

**Returns**

**deserialize** (*value*, *parent=None*)  
 Deserialize python object value to TelegramObject value

**TextField**

**class** aiogram.types.fields.**TextField**(\*, *prefix=None*, *suffix=None*, *default=None*,  
*alias=None*)  
 Bases: *aiogram.types.fields.Field*

Init prop

**Parameters**

- **base** – class for child element
- **default** – default value
- **alias** – alias name (for e.g. field 'from' has to be named 'from\_user' as 'from' is a builtin Python keyword)
- **on\_change** – callback will be called when value is changed

**serialize** (*value*)  
 Serialize value to python

**Parameters value** –

**Returns**

**deserialize** (*value*, *parent=None*)  
 Deserialize python object value to TelegramObject value

**Mixins****Downloadable**

**class** aiogram.types.mixins.**Downloadable**  
 Bases: object

Mixin for files

**async download** (*destination=None*, *timeout=30*, *chunk\_size=65536*, *seek=True*, *make\_dirs=True*)  
 Download file

**Parameters**

- **destination** – filename or instance of `io.IOBase`. For e. g. `io.BytesIO`
- **timeout** – Integer
- **chunk\_size** – Integer
- **seek** – Boolean - go to start of file when downloading is finished.
- **make\_dirs** – Make dirs if not exist

**Returns** destination

**async get\_file** ()  
 Get file information

**Returns** `aiogram.types.File`

**async get\_url()**

Get file url.

Attention!! This method has security vulnerabilities for the reason that result contains bot's *access token* in open form. Use at your own risk!

**Returns** url

## Types

### StickerSet

**class** `aiogram.types.sticker_set.StickerSet` (*conf: Optional[Dict[str, Any]] = None, \*\*kwargs: Any*)

Bases: `aiogram.types.base.TelegramObject`

This object represents a sticker set.

<https://core.telegram.org/bots/api#stickerset>

Deserialize object

#### Parameters

- **conf** –
- **kwargs** –

### EncryptedCredentials

**class** `aiogram.types.encrypted_credentials.EncryptedCredentials` (*conf: Optional[Dict[str, Any]] = None, \*\*kwargs: Any*)

Bases: `aiogram.types.base.TelegramObject`

Contains data required for decrypting and authenticating `EncryptedPassportElement`. See the Telegram Passport Documentation for a complete description of the data decryption and authentication processes.

<https://core.telegram.org/bots/api#encryptedcredentials>

Deserialize object

#### Parameters

- **conf** –
- **kwargs** –

## CallbackQuery

**class** aiogram.types.callback\_query.**CallbackQuery** (*conf: Optional[Dict[str, Any]] = None, \*\*kwargs: Any*)

Bases: *aiogram.types.base.TelegramObject*

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.

<https://core.telegram.org/bots/api#callbackquery>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

**async answer** (*text: Optional[String] = None, show\_alert: Optional[Boolean] = None, url: Optional[String] = None, cache\_time: Optional[Integer] = None*)

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

Alternatively, the user can be redirected to the specified Game URL. For this option to work, you must first create a game for your bot via @Botfather and accept the terms. Otherwise, you may use links like `t.me/your_bot?start=XXXX` that open your bot with a parameter.

Source: <https://core.telegram.org/bots/api#answercallbackquery>

### Parameters

- **text** (*typing.Optional[base.String]*) – Text of the notification. If not specified, nothing will be shown to the user, 0-200 characters
- **show\_alert** (*typing.Optional[base.Boolean]*) – If true, an alert will be shown by the client instead of a notification at the top of the chat screen. Defaults to false.
- **url** (*typing.Optional[base.String]*) – URL that will be opened by the user's client.
- **cache\_time** (*typing.Optional[base.Integer]*) – The maximum amount of time in seconds that the result of the callback query may be cached client-side.

**Returns** On success, True is returned.

**Return type** `base.Boolean`

## SuccessfulPayment

```
class aiogram.types.successful_payment.SuccessfulPayment (conf: Optional[Dict[str, Any]] = None, **kwargs: Any)
```

Bases: *aiogram.types.base.TelegramObject*

This object contains basic information about a successful payment.

<https://core.telegram.org/bots/api#successfulpayment>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

## MessageEntity

```
class aiogram.types.message_entity.MessageEntity (type: String, offset: Integer, length: Integer, url: String = None, user: aiogram.types.user.User = None, language: String = None, **kwargs)
```

Bases: *aiogram.types.base.TelegramObject*

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

<https://core.telegram.org/bots/api#messageentity>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

**get\_text** (text)

Get value of entity

**Parameters** **text** – full text

**Returns** part of text

**parse** (text, as\_html=True)

Get entity value with markup

### Parameters

- **text** – original text
- **as\_html** – as html?

**Returns** entity text with markup

## MessageEntityType

**class** aiogram.types.message\_entity.**MessageEntityType**

Bases: aiogram.utils.helper.Helper

List of entity types

**Key** MENTION

**Key** HASHTAG

**Key** CASHTAG

**Key** BOT\_COMMAND

**Key** URL

**Key** EMAIL

**Key** PHONE\_NUMBER

**Key** BOLD

**Key** ITALIC

**Key** CODE

**Key** PRE

**Key** UNDERLINE

**Key** STRIKETHROUGH

**Key** TEXT\_LINK

**Key** TEXT\_MENTION

## ShippingQuery

**class** aiogram.types.shipping\_query.**ShippingQuery** (*conf: Optional[Dict[str, Any]] = None, \*\*kwargs: Any*)

Bases: *aiogram.types.base.TelegramObject*

This object contains information about an incoming shipping query.

<https://core.telegram.org/bots/api#shippingquery>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

## PassportData

**class** aiogram.types.passport\_data.**PassportData** (*conf: Optional[Dict[str, Any]] = None, \*\*kwargs: Any*)

Bases: *aiogram.types.base.TelegramObject*

Contains information about Telegram Passport data shared with the bot by the user.

<https://core.telegram.org/bots/api#passportdata>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

## InlineKeyboardMarkup

**class** aiogram.types.inline\_keyboard.**InlineKeyboardMarkup** (*row\_width=3, inline\_keyboard=None, \*\*kwargs*)

Bases: *aiogram.types.base.TelegramObject*

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

Note: This will only work in Telegram versions released after 9 April, 2016. Older clients will display unsupported message.

<https://core.telegram.org/bots/api#inlinekeyboardmarkup>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

**add** (*\*args*)

Add buttons

**Parameters** **args** –

**Returns** self

**Return type** types.InlineKeyboardMarkup

**row** (*\*args*)

Add row

**Parameters** **args** –

**Returns** self

**Return type** types.InlineKeyboardMarkup

**insert** (*button*)

Insert button to last row

**Parameters** **button** –

**Returns** self



**Return type** `types.InlineKeyboardMarkup`

## InlineKeyboardButton

```
class aiogram.types.inline_keyboard.InlineKeyboardButton (text: String, url: String
                                                         = None, login_url:
                                                         aiogram.types.login_url.LoginUrl
                                                         = None, callback_data:
                                                         String = None,
                                                         switch_inline_query:
                                                         String = None,
                                                         switch_inline_query_current_chat:
                                                         String = None,
                                                         callback_game:
                                                         aiogram.types.callback_game.CallbackGame
                                                         = None, pay: Boolean =
                                                         None, **kwargs)
```

Bases: `aiogram.types.base.TelegramObject`

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

<https://core.telegram.org/bots/api#inlinekeyboardbutton>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

## User

```
class aiogram.types.user.User (conf: Optional[Dict[str, Any]] = None, **kwargs: Any)
```

Bases: `aiogram.types.base.TelegramObject`

This object represents a Telegram user or bot.

<https://core.telegram.org/bots/api#user>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

### property full\_name

You can get full name of user.

**Returns** str

### property mention

You can get user's username to mention him Full name will be returned if user has no username

**Returns** str

### property locale

Get user's locale

**Returns** `babel.core.Locale`

## Video

**class** `aiogram.types.video.Video` (*conf: Optional[Dict[str, Any]] = None, \*\*kwargs: Any*)  
Bases: `aiogram.types.base.TelegramObject`, `aiogram.types.mixins.Downloadable`

This object represents a video file.

<https://core.telegram.org/bots/api#video>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

## EncryptedPassportElement

**class** `aiogram.types.encrypted_passport_element.EncryptedPassportElement` (*conf: Optional[Dict[str, Any]] = None, \*\*kwargs: Any*)  
Bases: `aiogram.types.base.TelegramObject`

Contains information about documents or other Telegram Passport elements shared with the bot by the user.

<https://core.telegram.org/bots/api#encryptedpassportelement>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

## Game

**class** `aiogram.types.game.Game` (*conf: Optional[Dict[str, Any]] = None, \*\*kwargs: Any*)  
Bases: `aiogram.types.base.TelegramObject`

This object represents a game.

Use BotFather to create and edit games, their short names will act as unique identifiers.

<https://core.telegram.org/bots/api#game>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

## File

**class** aiogram.types.file.**File** (*conf: Optional[Dict[str, Any]] = None, \*\*kwargs: Any*)  
 Bases: *aiogram.types.base.TelegramObject, aiogram.types.mixins.Downloadable*

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>](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`.

Maximum file size to download is 20 MB

<https://core.telegram.org/bots/api#file>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

## LabeledPrice

**class** aiogram.types.labeled\_price.**LabeledPrice** (*label: String, amount: Integer*)  
 Bases: *aiogram.types.base.TelegramObject*

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

<https://core.telegram.org/bots/api#labeledprice>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

## CallbackGame

**class** aiogram.types.callback\_game.**CallbackGame** (*conf: Optional[Dict[str, Any]] = None, \*\*kwargs: Any*)

Bases: *aiogram.types.base.TelegramObject*

A placeholder, currently holds no information. Use BotFather to set up your game.

<https://core.telegram.org/bots/api#callbackgame>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

## ReplyKeyboardMarkup

```
class aiogram.types.reply_keyboard.ReplyKeyboardMarkup (keyboard: Optional[List[List[aiogram.types.reply_keyboard.KeyboardButton]]] = None, resize_keyboard: Boolean = None, one_time_keyboard: Boolean = None, selective: Boolean = None, row_width: Integer = 3)
```

Bases: *aiogram.types.base.TelegramObject*

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

<https://core.telegram.org/bots/api#replykeyboardmarkup>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

**add** (\*args)

Add buttons

**Parameters** args –

**Returns** self

**Return type** types.ReplyKeyboardMarkup

**row** (\*args)

Add row

**Parameters** args –

**Returns** self

**Return type** types.ReplyKeyboardMarkup

**insert** (button)

Insert button to last row

**Parameters** button –

**Returns** self

**Return type** types.ReplyKeyboardMarkup

## KeyboardButton

```
class aiogram.types.reply_keyboard.KeyboardButton (text: String, request_contact: Boolean = None, request_location: Boolean = None, request_poll: aiogram.types.reply_keyboard.KeyboardButtonPollType = None, **kwargs)
```

Bases: *aiogram.types.base.TelegramObject*

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 request\_contact, request\_location, and request\_poll are

mutually exclusive. Note: `request_contact` and `request_location` options will only work in Telegram versions released after 9 April, 2016. Older clients will ignore them. Note: `request_poll` option will only work in Telegram versions released after 23 January, 2020. Older clients will receive unsupported message.

<https://core.telegram.org/bots/api#keyboardbutton>

Deserialize object

#### Parameters

- **conf** –
- **kwargs** –

## ReplyKeyboardRemove

**class** `aiogram.types.reply_keyboard.ReplyKeyboardRemove` (*selective: Boolean = None*)

Bases: `aiogram.types.base.TelegramObject`

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`).

<https://core.telegram.org/bots/api#replykeyboardremove>

Deserialize object

#### Parameters

- **conf** –
- **kwargs** –

## Chat

**class** `aiogram.types.chat.Chat` (*conf: Optional[Dict[str, Any]] = None, \*\*kwargs: Any*)

Bases: `aiogram.types.base.TelegramObject`

This object represents a chat.

<https://core.telegram.org/bots/api#chat>

Deserialize object

#### Parameters

- **conf** –
- **kwargs** –

#### property mention

Get mention if a Chat has a username, or get full name if this is a Private Chat, otherwise None is returned

#### property shifted\_id

Get shifted id of chat, e.g. for private links

For example: -1001122334455 -> 1122334455

#### async get\_url () → String

Use this method to get chat link. Private chat returns user link. Other chat types return either username link (if they are public) or invite link (if they are private). :return: link :rtype: `base.String`

**async update\_chat** ()

Use this method to update Chat data

**Returns** None

**async set\_photo** (*photo*: `aiogram.types.input_file.InputFile`) → Boolean

Use this method to set a new profile photo for the chat. Photos can't be changed for private chats. The bot must be an administrator in the chat for this to work and must have the appropriate admin rights.

Note: In regular groups (non-supergroups), this method will only work if the 'All Members Are Admins' setting is off in the target group.

Source: <https://core.telegram.org/bots/api#setchatphoto>

**Parameters** **photo** (`base.InputFile`) – New chat photo, uploaded using multipart/form-data

**Returns** Returns True on success.

**Return type** `base.Boolean`

**async delete\_photo** () → Boolean

Use this method to delete a chat photo. Photos can't be changed for private chats. The bot must be an administrator in the chat for this to work and must have the appropriate admin rights.

Note: In regular groups (non-supergroups), this method will only work if the 'All Members Are Admins' setting is off in the target group.

Source: <https://core.telegram.org/bots/api#deletechatphoto>

**Returns** Returns True on success.

**Return type** `base.Boolean`

**async set\_title** (*title*: `String`) → Boolean

Use this method to change the title of a chat. Titles can't be changed for private chats. The bot must be an administrator in the chat for this to work and must have the appropriate admin rights.

Note: In regular groups (non-supergroups), this method will only work if the 'All Members Are Admins' setting is off in the target group.

Source: <https://core.telegram.org/bots/api#setchattitle>

**Parameters** **title** (`base.String`) – New chat title, 1-255 characters

**Returns** Returns True on success.

**Return type** `base.Boolean`

**async set\_description** (*description*: `String`) → Boolean

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

Source: <https://core.telegram.org/bots/api#setchatdescription>

**Parameters** **description** (`typing.Optional[base.String]`) – New chat description, 0-255 characters

**Returns** Returns True on success.

**Return type** `base.Boolean`

**async kick** (*user\_id*: `Integer`, *until\_date*: `Optional[Union[Integer, datetime.datetime, datetime.timedelta]] = None`) → Boolean

Use this method to kick a user from a group, a supergroup or a channel. In the case of supergroups

and channels, the user will not be able to return to the group on their own using invite links, etc., unless unbanned first.

The bot must be an administrator in the chat for this to work and must have the appropriate admin rights.

Note: In regular groups (non-supergroups), this method will only work if the ‘All Members Are Admins’ setting is off in the target group. Otherwise members may only be removed by the group’s creator or by the member that added them.

Source: <https://core.telegram.org/bots/api#kickchatmember>

#### Parameters

- **user\_id** (`base.Integer`) – Unique identifier of the target user
- **until\_date** (`typing.Optional[base.Integer]`) – Date when the user will be unbanned, unix time.

**Returns** Returns True on success.

**Return type** `base.Boolean`

**async unban** (*user\_id: Integer, only\_if\_banned: Optional[Boolean] = None*) → Boolean

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

Source: <https://core.telegram.org/bots/api#unbanchatmember>

#### Parameters

- **user\_id** (`base.Integer`) – Unique identifier of the target user
- **only\_if\_banned** (`typing.Optional[base.Boolean]`) – Do nothing if the user is not banned

**Returns** Returns True on success.

**Return type** `base.Boolean`

**async restrict** (*user\_id: Integer, permissions: Optional[aiogram.types.chat\_permissions.ChatPermissions] = None, until\_date: Optional[Union(Integer, datetime.datetime, datetime.timedelta)] = None, can\_send\_messages: Optional[Boolean] = None, can\_send\_media\_messages: Optional[Boolean] = None, can\_send\_other\_messages: Optional[Boolean] = None, can\_add\_web\_page\_previews: Optional[Boolean] = None*) → Boolean

Use this method to restrict a user in a supergroup. The bot must be an administrator in the supergroup for this to work and must have the appropriate admin rights. Pass True for all boolean parameters to lift restrictions from a user.

Source: <https://core.telegram.org/bots/api#restrictchatmember>

#### Parameters

- **user\_id** (`base.Integer`) – Unique identifier of the target user
- **permissions** (`ChatPermissions`) – New user permissions
- **until\_date** (`typing.Optional[base.Integer]`) – Date when restrictions will be lifted for the user, unix time.
- **can\_send\_messages** (`typing.Optional[base.Boolean]`) – Pass True, if the user can send text messages, contacts, locations and venues

- **can\_send\_media\_messages** (`typing.Optional[base.Boolean]`) – Pass True, if the user can send audios, documents, photos, videos, video notes and voice notes, implies `can_send_messages`
- **can\_send\_other\_messages** (`typing.Optional[base.Boolean]`) – Pass True, if the user can send animations, games, stickers and use inline bots, implies `can_send_media_messages`
- **can\_add\_web\_page\_previews** (`typing.Optional[base.Boolean]`) – Pass True, if the user may add web page previews to their messages, implies `can_send_media_messages`

**Returns** Returns True on success.

**Return type** `base.Boolean`

**async promote** (*user\_id: Integer, is\_anonymous: Optional[Boolean] = None, can\_change\_info: Optional[Boolean] = None, can\_post\_messages: Optional[Boolean] = None, can\_edit\_messages: Optional[Boolean] = None, can\_delete\_messages: Optional[Boolean] = None, can\_invite\_users: Optional[Boolean] = None, can\_restrict\_members: Optional[Boolean] = None, can\_pin\_messages: Optional[Boolean] = None, can\_promote\_members: Optional[Boolean] = None*) → `Boolean`

Use this method to promote or demote a user in a supergroup or a channel. The bot must be an administrator in the chat for this to work and must have the appropriate admin rights. Pass False for all boolean parameters to demote a user.

Source: <https://core.telegram.org/bots/api#promotechatmember>

#### Parameters

- **user\_id** (`base.Integer`) – Unique identifier of the target user
- **is\_anonymous** (`typing.Optional[base.Boolean]`) – Pass True, if the administrator's presence in the chat is hidden
- **can\_change\_info** (`typing.Optional[base.Boolean]`) – Pass True, if the administrator can change chat title, photo and other settings
- **can\_post\_messages** (`typing.Optional[base.Boolean]`) – Pass True, if the administrator can create channel posts, channels only
- **can\_edit\_messages** (`typing.Optional[base.Boolean]`) – Pass True, if the administrator can edit messages of other users, channels only
- **can\_delete\_messages** (`typing.Optional[base.Boolean]`) – Pass True, if the administrator can delete messages of other users
- **can\_invite\_users** (`typing.Optional[base.Boolean]`) – Pass True, if the administrator can invite new users to the chat
- **can\_restrict\_members** (`typing.Optional[base.Boolean]`) – Pass True, if the administrator can restrict, ban or unban chat members
- **can\_pin\_messages** (`typing.Optional[base.Boolean]`) – Pass True, if the administrator can pin messages, supergroups only
- **can\_promote\_members** (`typing.Optional[base.Boolean]`) – Pass True, if the administrator can add new administrators with a subset of his own privileges or demote administrators that he has promoted, directly or indirectly (promoted by administrators that were appointed by him)

**Returns** Returns True on success.



**Return type** `base.Boolean`

**async set\_permissions** (*permissions:* `aiogram.types.chat_permissions.ChatPermissions`) → `Boolean`

Use this method to set default chat permissions for all members. The bot must be an administrator in the group or a supergroup for this to work and must have the `can_restrict_members` admin rights.

Returns True on success.

**Parameters** `permissions` – New default chat permissions

**Returns** True on success.

**async set\_administrator\_custom\_title** (*user\_id:* `Integer`, *custom\_title:* `String`) → `Boolean`

Use this method to set a custom title for an administrator in a supergroup promoted by the bot.

Returns True on success.

Source: <https://core.telegram.org/bots/api#setchatadministratorcustomtitle>

**Parameters**

- **user\_id** – Unique identifier of the target user
- **custom\_title** – New custom title for the administrator; 0-16 characters, emoji are not allowed

**Returns** True on success.

**async pin\_message** (*message\_id:* `Integer`, *disable\_notification:* `Optional[Boolean] = False`) → `Boolean`

Use this method to add a message to the list of pinned messages in a chat. If the chat is not a private chat, the bot must be an administrator in the chat for this to work and must have the ‘`can_pin_messages`’ admin right in a supergroup or ‘`can_edit_messages`’ admin right in a channel. Returns True on success.

Source: <https://core.telegram.org/bots/api#pinchatmessage>

**Parameters**

- **message\_id** (`base.Integer`) – Identifier of a message to pin
- **disable\_notification** (`typing.Optional[base.Boolean]`) – Pass True, if it is not necessary to send a notification to all group members about the new pinned message

**Returns** Returns True on success

**Return type** `base.Boolean`

**async unpin\_message** (*message\_id:* `Optional[Integer] = None`) → `Boolean`

Use this method to remove a message from the list of pinned messages in a chat. If the chat is not a private chat, the bot must be an administrator in the chat for this to work and must have the ‘`can_pin_messages`’ admin right in a supergroup or ‘`can_edit_messages`’ admin right in a channel. Returns True on success.

Source: <https://core.telegram.org/bots/api#unpinchatmessage>

**Parameters** **message\_id** (`typing.Optional[base.Integer]`) – Identifier of a message to unpin. If not specified, the most recent pinned message (by sending date) will be unpinned.

**Returns** Returns True on success

**Return type** `base.Boolean`

**async unpin\_all\_messages()**

Use this method to clear the list of pinned messages in a chat. If the chat is not a private chat, the bot must be an administrator in the chat for this to work and must have the 'can\_pin\_messages' admin right in a supergroup or 'can\_edit\_messages' admin right in a channel. Returns True on success.

Source: <https://core.telegram.org/bots/api#unpinallchatmessages>

**Returns** Returns True on success

**Return type** `base.Boolean`

**async leave()** → `Boolean`

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

Source: <https://core.telegram.org/bots/api#leavechat>

**Returns** Returns True on success.

**Return type** `base.Boolean`

**async get\_administrators()** → `List[aiogram.types.chat_member.ChatMember]`

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

Source: <https://core.telegram.org/bots/api#getchatadministrators>

**Returns** On success, returns an Array of ChatMember objects that contains information about all chat administrators except other bots. If the chat is a group or a supergroup and no administrators were appointed, only the creator will be returned.

**Return type** `typing.List[types.ChatMember]`

**async get\_members\_count()** → `Integer`

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

Source: <https://core.telegram.org/bots/api#getchatmemberscount>

**Returns** Returns Int on success.

**Return type** `base.Integer`

**async get\_member(user\_id: Integer)** → `aiogram.types.chat_member.ChatMember`

Use this method to get information about a member of a chat.

Source: <https://core.telegram.org/bots/api#getchatmember>

**Parameters** **user\_id** (`base.Integer`) – Unique identifier of the target user

**Returns** Returns a ChatMember object on success.

**Return type** `types.ChatMember`

**async set\_sticker\_set(sticker\_set\_name: String)** → `Boolean`

Use this method to set a new group sticker set for a supergroup. The bot must be an administrator in the chat for this to work and must have the appropriate admin rights.

Use the field `can_set_sticker_set` optionally returned in `getChat` requests to check if the bot can use this method.

Source: <https://core.telegram.org/bots/api#setchatstickerset>

**Parameters** **sticker\_set\_name** (`base.String`) – Name of the sticker set to be set as the group sticker set

**Returns** Returns True on success

**Return type** `base.Boolean`

**async delete\_sticker\_set()** → Boolean

Use this method to delete a group sticker set from a supergroup. The bot must be an administrator in the chat for this to work and must have the appropriate admin rights.

Use the field `can_set_sticker_set` optionally returned in `getChat` requests to check if the bot can use this method.

Source: <https://core.telegram.org/bots/api#deletechatstickerset>

**Returns** Returns True on success

**Return type** `base.Boolean`

**async do(action: String)** → Boolean

Use this method when you need to tell the user that something is happening on the bot's side. The status is set for 5 seconds or less (when a message arrives from your bot, Telegram clients clear its typing status).

We only recommend using this method when a response from the bot will take a noticeable amount of time to arrive.

Source: <https://core.telegram.org/bots/api#sendchataction>

**Parameters** `action` (`base.String`) – Type of action to broadcast.

**Returns** Returns True on success.

**Return type** `base.Boolean`

**async export\_invite\_link()** → String

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

Source: <https://core.telegram.org/bots/api#exportchatinvitelink>

**Returns** Returns exported invite link as String on success.

**Return type** `base.String`

## ChatType

**class** `aiogram.types.chat.ChatType`

Bases: `aiogram.utils.helper.Helper`

List of chat types

**Key** `PRIVATE`

**Key** `GROUP`

**Key** `SUPER_GROUP`

**Key** `SUPERGROUP`

**Key** `CHANNEL`

**classmethod** `is_private(obj)` → bool

Check chat is private

**Parameters** `obj` –

**Returns**

**classmethod** `is_group(obj)` → bool

Check chat is group

**Parameters** *obj* –

**Returns**

**classmethod** **is\_super\_group** (*obj*) → bool

Check chat is super-group

**Parameters** *obj* –

**Returns**

**classmethod** **is\_group\_or\_super\_group** (*obj*) → bool

Check chat is group or super-group

**Parameters** *obj* –

**Returns**

**classmethod** **is\_channel** (*obj*) → bool

Check chat is channel

**Parameters** *obj* –

**Returns**

## ChatActions

**class** aiogram.types.chat.ChatActions

Bases: aiogram.utils.helper.Helper

List of chat actions

**Key** TYPING

**Key** UPLOAD\_PHOTO

**Key** RECORD\_VIDEO

**Key** UPLOAD\_VIDEO

**Key** RECORD\_AUDIO

**Key** UPLOAD\_AUDIO

**Key** UPLOAD\_DOCUMENT

**Key** FIND\_LOCATION

**Key** RECORD\_VIDEO\_NOTE

**Key** UPLOAD\_VIDEO\_NOTE

**classmethod** **calc\_timeout** (*text*, *timeout=0.8*)

Calculate timeout for text

**Parameters**

• **text** –

• **timeout** –

**Returns**

**async classmethod** **typing** (*sleep=None*)

Do typing

**Parameters** **sleep** – sleep timeout

**Returns****async classmethod upload\_photo** (*sleep=None*)

Do upload\_photo

**Parameters** **sleep** – sleep timeout**Returns****async classmethod record\_video** (*sleep=None*)

Do record video

**Parameters** **sleep** – sleep timeout**Returns****async classmethod upload\_video** (*sleep=None*)

Do upload video

**Parameters** **sleep** – sleep timeout**Returns****async classmethod record\_audio** (*sleep=None*)

Do record audio

**Parameters** **sleep** – sleep timeout**Returns****async classmethod upload\_audio** (*sleep=None*)

Do upload audio

**Parameters** **sleep** – sleep timeout**Returns****async classmethod upload\_document** (*sleep=None*)

Do upload document

**Parameters** **sleep** – sleep timeout**Returns****async classmethod find\_location** (*sleep=None*)

Do find location

**Parameters** **sleep** – sleep timeout**Returns****async classmethod record\_video\_note** (*sleep=None*)

Do record video note

**Parameters** **sleep** – sleep timeout**Returns****async classmethod upload\_video\_note** (*sleep=None*)

Do upload video note

**Parameters** **sleep** – sleep timeout**Returns**

## Document

```
class aiogram.types.document.Document (conf: Optional[Dict[str, Any]] = None, **kwargs: Any)
```

Bases: *aiogram.types.base.TelegramObject, aiogram.types.mixins.Downloadable*

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

<https://core.telegram.org/bots/api#document>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

## Audio

```
class aiogram.types.audio.Audio (conf: Optional[Dict[str, Any]] = None, **kwargs: Any)
```

Bases: *aiogram.types.base.TelegramObject, aiogram.types.mixins.Downloadable*

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

<https://core.telegram.org/bots/api#audio>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

## ForceReply

```
class aiogram.types.force_reply.ForceReply (conf: Optional[Dict[str, Any]] = None, **kwargs: Any)
```

Bases: *aiogram.types.base.TelegramObject*

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.

Example: A poll bot for groups runs in privacy mode (only receives commands, replies to its messages and mentions). There could be two ways to create a new poll

The last option is definitely more attractive. And if you use ForceReply in your bot's questions, it will receive the user's answers even if it only receives replies, commands and mentions — without any extra work for the user.

<https://core.telegram.org/bots/api#forcereply>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

**classmethod create** (*selective: Optional[Boolean] = None*)

Create new force reply

**Parameters** **selective** –

**Returns**

## PassportElementError

```
class aiogram.types.passport_element_error.PassportElementError (conf: Optional[Dict[str, Any]] = None, **kwargs: Any)
```

Bases: *aiogram.types.base.TelegramObject*

This object represents an error in the Telegram Passport element which was submitted that should be resolved by the user.

<https://core.telegram.org/bots/api#passportelementerror>

Deserialize object

**Parameters**

- **conf** –
- **kwargs** –

## PassportElementErrorDataField

```
class aiogram.types.passport_element_error.PassportElementErrorDataField (source: String, type: String, field_name: String, data_hash: String, message: String)
```

Bases: *aiogram.types.passport\_element\_error.PassportElementError*

Represents an issue in one of the data fields that was provided by the user. The error is considered resolved when the field's value changes.

<https://core.telegram.org/bots/api#passportelementerrordatafield>

Deserialize object

**Parameters**

- **conf** –
- **kwargs** –

## PassportElementErrorFile

```
class aiogram.types.passport_element_error.PassportElementErrorFile (source:  
                                                                    String,  
                                                                    type:  
                                                                    String,  
                                                                    file_hash:  
                                                                    String,  
                                                                    mes-  
                                                                    sage:  
                                                                    String)
```

Bases: *aiogram.types.passport\_element\_error.PassportElementError*

Represents an issue with a document scan. The error is considered resolved when the file with the document scan changes.

<https://core.telegram.org/bots/api#passportelementerrorfile>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

## PassportElementErrorFiles

```
class aiogram.types.passport_element_error.PassportElementErrorFiles (source:  
                                                                    String,  
                                                                    type:  
                                                                    String,  
                                                                    file_hashes:  
                                                                    List[String],  
                                                                    mes-  
                                                                    sage:  
                                                                    String)
```

Bases: *aiogram.types.passport\_element\_error.PassportElementError*

Represents an issue with a list of scans. The error is considered resolved when the list of files containing the scans changes.

<https://core.telegram.org/bots/api#passportelementerrorfiles>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –



## PassportElementErrorFrontSide

```
class aiogram.types.passport_element_error.PassportElementErrorFrontSide (source:
    String,
    type:
    String,
    file_hash:
    String,
    mes-
    sage:
    String)
```

Bases: *aiogram.types.passport\_element\_error.PassportElementError*

Represents an issue with the front side of a document. The error is considered resolved when the file with the front side of the document changes.

<https://core.telegram.org/bots/api#passportelementerrorfrontside>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

## PassportElementErrorReverseSide

```
class aiogram.types.passport_element_error.PassportElementErrorReverseSide (source:
    String,
    type:
    String,
    file_hash:
    String,
    mes-
    sage:
    String)
```

Bases: *aiogram.types.passport\_element\_error.PassportElementError*

Represents an issue with the reverse side of a document. The error is considered resolved when the file with reverse side of the document changes.

<https://core.telegram.org/bots/api#passportelementerrorreverseside>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

## PassportElementErrorSelfie

```
class aiogram.types.passport_element_error.PassportElementErrorSelfie (source:
                                                                    String,
                                                                    type:
                                                                    String,
                                                                    file_hash:
                                                                    String,
                                                                    mes-
                                                                    sage:
                                                                    String)
```

Bases: *aiogram.types.passport\_element\_error.PassportElementError*

Represents an issue with the selfie with a document. The error is considered resolved when the file with the selfie changes.

<https://core.telegram.org/bots/api#passportelementerrorselfie>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

## ShippingAddress

```
class aiogram.types.shipping_address.ShippingAddress (conf: Optional[Dict[str, Any]]
                                                         = None, **kwargs: Any)
```

Bases: *aiogram.types.base.TelegramObject*

This object represents a shipping address.

<https://core.telegram.org/bots/api#shippingaddress>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

## ResponseParameters

```
class aiogram.types.response_parameters.ResponseParameters (conf: Optional[Dict[str, Any]]
                                                                = None, **kwargs: Any)
```

Bases: *aiogram.types.base.TelegramObject*

Contains information about why a request was unsuccessful.

<https://core.telegram.org/bots/api#responseparameters>

Deserialize object

### Parameters

- **conf** –

- **kwargs** –

## OrderInfo

```
class aiogram.types.order_info.OrderInfo (conf: Optional[Dict[str, Any]] = None,
                                           **kwargs: Any)
```

Bases: `aiogram.types.base.TelegramObject`

This object represents information about an order.

<https://core.telegram.org/bots/api#orderinfo>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

## GameHighScore

```
class aiogram.types.game_high_score.GameHighScore (conf: Optional[Dict[str, Any]] =
                                                    None, **kwargs: Any)
```

Bases: `aiogram.types.base.TelegramObject`

This object represents one row of the high scores table for a game. And that's about all we've got for now. If you've got any questions, please check out our Bot FAQ

<https://core.telegram.org/bots/api#gamehighscore>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

## Sticker

```
class aiogram.types.sticker.Sticker (conf: Optional[Dict[str, Any]] = None, **kwargs: Any)
Bases: aiogram.types.base.TelegramObject, aiogram.types.mixins.Downloadable
```

This object represents a sticker.

<https://core.telegram.org/bots/api#sticker>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

**async set\_position\_in\_set** (*position: Integer*) → Boolean

Use this method to move a sticker in a set created by the bot to a specific position.

Source: <https://core.telegram.org/bots/api#setstickerpositioninset>

**Parameters** **position** (`base.Integer`) – New sticker position in the set, zero-based

**Returns** Returns True on success

**Return type** `base.Boolean`

**async delete\_from\_set** () → Boolean

Use this method to delete a sticker from a set created by the bot.

Source: <https://core.telegram.org/bots/api#deletestickerfromset>

**Parameters** **sticker** (`base.String`) – File identifier of the sticker

**Returns** Returns True on success

**Return type** `base.Boolean`

## InlineQuery

**class** `aiogram.types.inline_query.InlineQuery` (*conf: Optional[Dict[str, Any]] = None, \*\*kwargs: Any*)

Bases: `aiogram.types.base.TelegramObject`

This object represents an incoming inline query.

When the user sends an empty query, your bot could return some default or trending results.

<https://core.telegram.org/bots/api#inlinequery>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

**async answer** (*results: List[aiogram.types.inline\_query\_result.InlineQueryResult], cache\_time: Optional[Integer] = None, is\_personal: Optional[Boolean] = None, next\_offset: Optional[String] = None, switch\_pm\_text: Optional[String] = None, switch\_pm\_parameter: Optional[String] = None*)

Use this method to send answers to an inline query. No more than 50 results per query are allowed.

Source: <https://core.telegram.org/bots/api#answerinlinequery>

### Parameters

- **results** (`typing.List[types.InlineQueryResult]`) – A JSON-serialized array of results for the inline query
- **cache\_time** (`typing.Optional[base.Integer]`) – The maximum amount of time in seconds that the result of the inline query may be cached on the server. Defaults to 300.
- **is\_personal** (`typing.Optional[base.Boolean]`) – Pass True, if results may be cached on the server side only for the user that sent the query. By default, results may be returned to any user who sends the same query
- **next\_offset** (`typing.Optional[base.String]`) – Pass the offset that a client should send in the next query with the same text to receive more results. Pass an empty string if there are no more results or if you don't support pagination. Offset length can't exceed 64 bytes.
- **switch\_pm\_text** (`typing.Optional[base.String]`) – If passed, clients will display a button with specified text that switches the user to a private chat with the bot and sends the bot a start message with the parameter `switch_pm_parameter`

- **switch\_pm\_parameter** (`typing.Optional[base.String]`) – Deep-linking parameter for the /start message sent to the bot when user presses the switch button. 1-64 characters, only A-Z, a-z, 0-9, \_ and - are allowed.

**Returns** On success, True is returned

**Return type** `base.Boolean`

## Location

**class** `aiogram.types.location.Location` (`conf: Optional[Dict[str, Any]] = None, **kwargs: Any`)

Bases: `aiogram.types.base.TelegramObject`

This object represents a point on the map.

<https://core.telegram.org/bots/api#location>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

## Animation

**class** `aiogram.types.animation.Animation` (`conf: Optional[Dict[str, Any]] = None, **kwargs: Any`)

Bases: `aiogram.types.base.TelegramObject`, `aiogram.types.mixins.Downloadable`

You can provide an animation for your game so that it looks stylish in chats (check out Lumberjack for an example). This object represents an animation file to be displayed in the message containing a game.

<https://core.telegram.org/bots/api#animation>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

## InputMedia

**class** `aiogram.types.input_media.InputMedia` (`*args, **kwargs`)

Bases: `aiogram.types.base.TelegramObject`

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

- `InputMediaAnimation`
- `InputMediaDocument`
- `InputMediaAudio`
- `InputMediaPhoto`
- `InputMediaVideo`

That is only base class.

<https://core.telegram.org/bots/api#inputmedia>

Deserialize object

#### Parameters

- **conf** –
- **kwargs** –

### InputMediaAnimation

```
class aiogram.types.input_media.InputMediaAnimation(media: InputFile, thumb: Union[InputFile, String] = None, caption: String = None, width: Integer = None, height: Integer = None, duration: Integer = None, parse_mode: String = None, caption_entities: Optional[List[aiogram.types.message_entity.MessageEntity]] = None, **kwargs)
```

Bases: *aiogram.types.input\_media.InputMedia*

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

<https://core.telegram.org/bots/api#inputmediaanimation>

Deserialize object

#### Parameters

- **conf** –
- **kwargs** –

### InputMediaDocument

```
class aiogram.types.input_media.InputMediaDocument(media: InputFile, thumb: Optional[Union[InputFile, String]] = None, caption: String = None, parse_mode: String = None, caption_entities: Optional[List[aiogram.types.message_entity.MessageEntity]] = None, disable_content_type_detection: Optional[Boolean] = None, **kwargs)
```

Bases: *aiogram.types.input\_media.InputMedia*

Represents a general file to be sent.

<https://core.telegram.org/bots/api#inputmediadocument>

Deserialize object

#### Parameters

- **conf** –

- **kwargs** –

## InputMediaAudio

```
class aiogram.types.input_media.InputMediaAudio (media:      InputFile,      thumb:
                                                Union[InputFile, String] = None, cap-
tion: String = None, duration: Integer
= None, performer: String = None,
title: String = None, parse_mode:
String = None, caption_entities: Op-
tional[List[aiogram.types.message_entity.MessageEntity]]
= None, **kwargs)
```

Bases: *aiogram.types.input\_media.InputMedia*

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

<https://core.telegram.org/bots/api#inputmediaaudio>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

## InputMediaPhoto

```
class aiogram.types.input_media.InputMediaPhoto (media: InputFile, caption: String
= None, parse_mode: String
= None, caption_entities: Op-
tional[List[aiogram.types.message_entity.MessageEntity]]
= None, **kwargs)
```

Bases: *aiogram.types.input\_media.InputMedia*

Represents a photo to be sent.

<https://core.telegram.org/bots/api#inputmediaphoto>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

## InputMediaVideo

```
class aiogram.types.input_media.InputMediaVideo (media: InputFile, thumb: Union[InputFile, String] = None, caption: String = None, width: Integer = None, height: Integer = None, duration: Integer = None, parse_mode: String = None, caption_entities: Optional[List[aiogram.types.message_entity.MessageEntity]] = None, supports_streaming: Boolean = None, **kwargs)
```

Bases: *aiogram.types.input\_media.InputMedia*

Represents a video to be sent.

<https://core.telegram.org/bots/api#inputmediavideo>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

## MediaGroup

```
class aiogram.types.input_media.MediaGroup (medias: Optional[List[Union[aiogram.types.input_media.InputMedia, Dict]]] = None)
```

Bases: *aiogram.types.base.TelegramObject*

Helper for sending media group

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

**attach\_many** (\*medias: Union[aiogram.types.input\_media.InputMedia, Dict])

Attach list of media

### Parameters medias –

**attach** (media: Union[aiogram.types.input\_media.InputMedia, Dict])

Attach media

### Parameters media –

**attach\_photo** (photo: Union[aiogram.types.input\_media.InputMediaPhoto, InputFile], caption: String = None)

Attach photo

### Parameters

- **photo** –
- **caption** –



**attach\_video** (*video*: Union[aiogram.types.input\_media.InputMediaVideo, InputFile], *thumb*: Union[InputFile, String] = None, *caption*: String = None, *width*: Integer = None, *height*: Integer = None, *duration*: Integer = None)

Attach video

#### Parameters

- **video** –
- **caption** –
- **width** –
- **height** –
- **duration** –

**to\_python**() → List

Get object as JSON serializable

#### Returns

### InlineQueryResult

**class** aiogram.types.inline\_query\_result.**InlineQueryResult** (\*\*kwargs)

Bases: *aiogram.types.base.TelegramObject*

This object represents one result of an inline query.

Telegram clients currently support results of the following 20 types

<https://core.telegram.org/bots/api#inlinequeryresult>

Deserialize object

#### Parameters

- **conf** –
- **kwargs** –

## InlineQueryResultArticle

```
class aiogram.types.inline_query_result.InlineQueryResultArticle(*,  
    id: String, title: String, input_message_content: aiogram.types.input_message_content.  
    InputMessageContent, reply_markup: Optional[aiogram.types.inline_keyboard.  
    InlineKeyboardMarkup] = None, url: Optional[String] = None, hide_url: Op-  
    tional[Boolean] = None, description: Optional[String] = None, thumb_url: Op-  
    tional[String] = None, thumb_width: Optional[Integer] = None, thumb_height: Op-  
    tional[Integer] = None)
```

Bases: `aiogram.types.inline_query_result.InlineQueryResult`

Represents a link to an article or web page.

<https://core.telegram.org/bots/api#inlinequeryresultarticle>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

## InlineQueryResultPhoto

```

class aiogram.types.inline_query_result.InlineQueryResultPhoto(*, id: String,
                                                                photo_url:
                                                                String,
                                                                thumb_url:
                                                                String,
                                                                photo_width:
                                                                Optional[Integer]
                                                                = None,
                                                                photo_height:
                                                                Optional[Integer]
                                                                = None,
                                                                title: Optional[String]
                                                                = None,
                                                                description: Optional[String]
                                                                = None,
                                                                caption: Optional[String]
                                                                = None,
                                                                parse_mode:
                                                                Optional[String]
                                                                = None,
                                                                caption_entities:
                                                                Optional[List[aiogram.types.message_entity]]
                                                                = None,
                                                                reply_markup:
                                                                Optional[aiogram.types.inline_keyboard.InlineKeyboardMarkup]
                                                                = None,
                                                                input_message_content:
                                                                Optional[aiogram.types.input_message_content]
                                                                = None)

```

Bases: `aiogram.types.inline_query_result.InlineQueryResult`

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.

<https://core.telegram.org/bots/api#inlinequeryresultphoto>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

## InlineQueryResultGif

```
class aiogram.types.inline_query_result.InlineQueryResultGif(*id: String,  
    gif_url: String,  
    gif_width: Optional[Integer] = None,  
    gif_height: Optional[Integer] = None,  
    gif_duration: Optional[Integer] = None,  
    thumb_url: Optional[String] = None,  
    title: Optional[String] = None,  
    caption: Optional[String] = None,  
    parse_mode: Optional[String] = None,  
    reply_markup: Optional[aiogram.types.inline_keyboard.InlineKeyboardMarkup] = None,  
    caption_entities: Optional[List[aiogram.types.message_entity.MessageEntity]] = None,  
    input_message_content: Optional[aiogram.types.input_message_content.InputMessageContent] = None)
```

Bases: *aiogram.types.inline\_query\_result.InlineQueryResult*

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.

<https://core.telegram.org/bots/api#inlinequeryresultgif>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

## InlineQueryResultMpeg4Gif

```

class aiogram.types.inline_query_result.InlineQueryResultMpeg4Gif(*,      id:
                                                                    String,
                                                                    mpeg4_url:
                                                                    String,
                                                                    thumb_url:
                                                                    String,
                                                                    mpeg4_width:
                                                                    Optional[Integer]
                                                                    = None,
                                                                    mpeg4_height:
                                                                    Optional[Integer]
                                                                    = None,
                                                                    mpeg4_duration:
                                                                    Optional[Integer]
                                                                    = None,
                                                                    title: Optional[
                                                                    String]
                                                                    = None,
                                                                    caption:
                                                                    Optional[
                                                                    String]
                                                                    = None,
                                                                    parse_mode:
                                                                    Optional[
                                                                    String]
                                                                    = None,
                                                                    reply_markup:
                                                                    Optional[
                                                                    aiogram.types.inline_keyboard.InlineKeyboardMarkup]
                                                                    = None,
                                                                    caption_entities:
                                                                    Optional[
                                                                    List[aiogram.types.message_entities.MessageEntity]]
                                                                    = None,
                                                                    input_message_content:
                                                                    Optional[
                                                                    aiogram.types.input_message_content.InputMessageContent]
                                                                    = None)

```

Bases: `aiogram.types.inline_query_result.InlineQueryResult`

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.

<https://core.telegram.org/bots/api#inlinequeryresultmpeg4gif>

Deserialize object

### Parameters

- `conf` –
- `kwargs` –

## InlineQueryResultVideo

```
class aiogram.types.inline_query_result.InlineQueryResultVideo(*id: String,  
    video_url: String,  
    mime_type: String,  
    thumb_url: String,  
    title: String,  
    caption: Optional[String] = None,  
    parse_mode: Optional[String] = None,  
    video_width: Optional[Integer] = None,  
    video_height: Optional[Integer] = None,  
    video_duration: Optional[Integer] = None, description: Optional[String] = None, reply_markup: Optional[aiogram.types.inline_keyboard.InlineKeyboardMarkup] = None, caption_entities: Optional[List[aiogram.types.message_entity.MessageEntity]] = None, input_message_content: Optional[aiogram.types.input_message_content.InputMessageContent] = None)
```

Bases: `aiogram.types.inline_query_result.InlineQueryResult`

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.

If an `InlineQueryResultVideo` message contains an embedded video (e.g., YouTube), you must replace its content using `input_message_content`.

<https://core.telegram.org/bots/api#inlinequeryresultvideo>

Deserialize object

#### Parameters

- `conf` –
- `kwargs` –

### InlineQueryResultAudio

```
class aiogram.types.inline_query_result.InlineQueryResultAudio(*, id: String, audio_url: String, title: String, caption: Optional[String] = None, parse_mode: Optional[String] = None, performer: Optional[String] = None, audio_duration: Optional[Integer] = None, reply_markup: Optional[aiogram.types.inline_keyboard.InlineKeyboardMarkup] = None, caption_entities: Optional[List[aiogram.types.message_entity.MessageEntity]] = None, input_message_content: Optional[aiogram.types.input_message_content.InputMessageContent] = None)
```

Bases: `aiogram.types.inline_query_result.InlineQueryResult`

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.

Note: This will only work in Telegram versions released after 9 April, 2016. Older clients will ignore them.

<https://core.telegram.org/bots/api#inlinequeryresultaudio>

Deserialize object

#### Parameters

- `conf` –

- **kwargs** –

## InlineQueryResultVoice

```
class aiogram.types.inline_query_result.InlineQueryResultVoice(*id: String,  
                                                                voice_url:  
                                                                String,  
                                                                title: String,  
                                                                caption: Optional[String]  
                                                                = None,  
                                                                parse_mode:  
                                                                Optional[String]  
                                                                = None,  
                                                                voice_duration:  
                                                                Optional[Integer]  
                                                                = None, reply_markup:  
                                                                Optional[aiogram.types.inline_keyboard.InlineKeyboardMarkup]  
                                                                = None, caption_entities:  
                                                                Optional[List[aiogram.types.message_entity.MessageEntity]]  
                                                                = None, input_message_content:  
                                                                Optional[aiogram.types.input_message_content.InputMessageContent]  
                                                                = None)
```

Bases: `aiogram.types.inline_query_result.InlineQueryResult`

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.

Note: This will only work in Telegram versions released after 9 April, 2016. Older clients will ignore them.

<https://core.telegram.org/bots/api#inlinequeryresultvoice>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –



## InlineQueryResultDocument

```

class aiogram.types.inline_query_result.InlineQueryResultDocument(*,
    id:
        String, title:
        String, cap-
        tion: Op-
        tional[String]
        = None,
    parse_mode:
        Op-
        tional[String]
        = None,
    cap-
        tion_entities:
        Op-
        tional[List[aiogram.types.message_e
        = None,
    docu-
        ment_url:
        Op-
        tional[String]
        = None,
    mime_type:
        Op-
        tional[String]
        = None,
    descrip-
        tion: Op-
        tional[String]
        = None, re-
        ply_markup:
        Op-
        tional[aiogram.types.inline_keyboar
        = None, in-
        put_message_content:
        Op-
        tional[aiogram.types.input_message
        = None,
    thumb_url:
        Op-
        tional[String]
        = None,
    thumb_width:
        Op-
        tional[Integer]
        = None,
    thumb_height:
        Op-
        tional[Integer]
        = None)

```

Bases: `aiogram.types.inline_query_result.InlineQueryResult`

Represents a link to a file.

By default, this file will be sent by the user with an optional caption. Alternatively, you can use in-

`put_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.

Note: This will only work in Telegram versions released after 9 April, 2016. Older clients will ignore them.

<https://core.telegram.org/bots/api#inlinequeryresultdocument>

Deserialize object

**Parameters**

- **conf** –
- **kwargs** –

## InlineQueryResultLocation

```

class aiogram.types.inline_query_result.InlineQueryResultLocation(*,
    id:
        String, latitude:
        Float, longitude:
        Float, title:
        String, horizontal_accuracy:
        Optional[Float]
        = None, live_period:
        Optional[Integer]
        = None, heading:
        Optional[Integer]
        = None, proximity_alert_radius:
        Optional[Integer]
        = None, reply_markup:
        Optional[aiogram.types.inline_keyboard.InlineKeyboardMarkup]
        = None, input_message_content:
        Optional[aiogram.types.input_message_content.InputMessageContent]
        = None, thumb_url:
        Optional[String]
        = None, thumb_width:
        Optional[Integer]
        = None, thumb_height:
        Optional[Integer]
        = None)

```

Bases: `aiogram.types.inline_query_result.InlineQueryResult`

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.

<https://core.telegram.org/bots/api#inlinequeryresultlocation>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

### InlineQueryResultVenue

```
class aiogram.types.inline_query_result.InlineQueryResultVenue(*id: String,  
    latitude: Float,  
    longitude:  
    Float,    title:  
    String,    ad-  
    dress: String,  
    foursquare_id:  
    Op-  
    tional[String]  
    = None,  
    foursquare_type:  
    Op-  
    tional[String]  
    = None,  
    google_place_id:  
    Op-  
    tional[String]  
    = None,  
    google_place_type:  
    Op-  
    tional[String]  
    = None, re-  
    ply_markup:  
    Op-  
    tional[aiogram.types.inline\_keyboard.Inl]  
    = None, in-  
    put_message_content:  
    Op-  
    tional[aiogram.types.input\_message\_con]  
    = None,  
    thumb_url: Op-  
    tional[String]  
    = None,  
    thumb_width:  
    Op-  
    tional[Integer]  
    = None,  
    thumb_height:  
    Op-  
    tional[Integer]  
    = None)
```

Bases: [aiogram.types.inline\\_query\\_result.InlineQueryResult](#)

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.

Note: This will only work in Telegram versions released after 9 April, 2016. Older clients will ignore them.

<https://core.telegram.org/bots/api#inlinequeryresultvenue>

Deserialize object

#### Parameters

- **conf** –
- **kwargs** –

### InlineQueryResultContact

```
class aiogram.types.inline_query_result.InlineQueryResultContact (*, id: String,
                                                                    phone_number:
                                                                    String,
                                                                    first_name:
                                                                    String,
                                                                    last_name:
                                                                    Op-
                                                                    tional[String]
                                                                    = None, re-
                                                                    ply_markup:
                                                                    Op-
                                                                    tional[aiogram.types.inline_keyboard.
                                                                    = None, in-
                                                                    put_message_content:
                                                                    Op-
                                                                    tional[aiogram.types.input_message_c
                                                                    = None,
                                                                    thumb_url:
                                                                    Op-
                                                                    tional[String]
                                                                    = None,
                                                                    thumb_width:
                                                                    Op-
                                                                    tional[Integer]
                                                                    = None,
                                                                    thumb_height:
                                                                    Op-
                                                                    tional[Integer]
                                                                    = None,
                                                                    foursquare_type:
                                                                    Op-
                                                                    tional[String]
                                                                    = None)
```

Bases: `aiogram.types.inline_query_result.InlineQueryResult`

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.

Note: This will only work in Telegram versions released after 9 April, 2016. Older clients will ignore them.

<https://core.telegram.org/bots/api#inlinequeryresultcontact>

Deserialize object

#### Parameters

- **conf** –
- **kwargs** –

### InlineQueryResultGame

```
class aiogram.types.inline_query_result.InlineQueryResultGame(*, id: String,
                                                                game_short_name:
                                                                String, re-
                                                                ply_markup: Op-
                                                                tional[aiogram.types.inline_keyboard.Inlin
                                                                = None)
```

Bases: *aiogram.types.inline\_query\_result.InlineQueryResult*

Represents a Game.

Note: This will only work in Telegram versions released after October 1, 2016. Older clients will not display any inline results if a game result is among them.

<https://core.telegram.org/bots/api#inlinequeryresultgame>

Deserialize object

#### Parameters

- **conf** –
- **kwargs** –

## InlineQueryResultCachedPhoto

```

class aiogram.types.inline_query_result.InlineQueryResultCachedPhoto(*, id:
    String,
    photo_file_id:
    String,
    title:
    Optional[String]
    =
    None,
    description:
    Optional[String]
    =
    None,
    caption:
    Optional[String]
    =
    None,
    parse_mode:
    Optional[String]
    =
    None,
    caption_entities:
    Optional[List[aiogram.types.messa
    =
    None,
    reply_markup:
    Optional[aiogram.types.inline_key
    =
    None,
    input_message_content:
    Optional[aiogram.types.input_mes
    =
    None)

```

Bases: `aiogram.types.inline_query_result.InlineQueryResult`

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.

<https://core.telegram.org/bots/api#inlinequeryresultcachedphoto>

Deserialize object

#### Parameters

- **conf** –
- **kwargs** –

### InlineQueryResultCachedGif

```
class aiogram.types.inline_query_result.InlineQueryResultCachedGif(*,      id:
                                                                    String,
                                                                    gif_file_id:
                                                                    String,
                                                                    title: Op-
                                                                    tional[String]
                                                                    = None,
                                                                    cap-
                                                                    tion: Op-
                                                                    tional[String]
                                                                    = None,
                                                                    parse_mode:
                                                                    Op-
                                                                    tional[String]
                                                                    = None,
                                                                    cap-
                                                                    tion_entities:
                                                                    Op-
                                                                    tional[List[aiogram.types.message
                                                                    = None,
                                                                    re-
                                                                    ply_markup:
                                                                    Op-
                                                                    tional[aiogram.types.inline_keybo
                                                                    = None,
                                                                    in-
                                                                    put_message_content:
                                                                    Op-
                                                                    tional[aiogram.types.input_messag
                                                                    = None)
```

Bases: *aiogram.types.inline\_query\_result.InlineQueryResult*

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.

<https://core.telegram.org/bots/api#inlinequeryresultcachedgif>

Deserialize object

#### Parameters

- **conf** –
- **kwargs** –



## InlineQueryResultCachedMpeg4Gif

```

class aiogram.types.inline_query_result.InlineQueryResultCachedMpeg4Gif(*,
    id:
        String,
    mpeg4_file_id:
        String,
    title:
        Optional[String]
    =
        None,
    caption:
        Optional[String]
    =
        None,
    parse_mode:
        Optional[String]
    =
        None,
    caption_entities:
        Optional[List[aiogram.types.m
    =
        None,
    reply_markup:
        Optional[aiogram.types.inline
    =
        None,
    input_message_content:
        Optional[aiogram.types.input_
    =
        None)

```

Bases: *aiogram.types.inline\_query\_result.InlineQueryResult*

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.

<https://core.telegram.org/bots/api#inlinequeryresultcachedmpeg4gif>

Deserialize object

### Parameters

- `conf` –

- **kwargs** –

### InlineQueryResultCachedSticker

```
class aiogram.types.inline_query_result.InlineQueryResultCachedSticker(*,
    id:
    String,
    sticker_file_id:
    String,
    re-
    ply_markup:
    Op-
    tional[aiogram.types.inline_q-
    =
    None,
    in-
    put_message_content:
    Op-
    tional[aiogram.types.input_m-
    =
    None)
```

Bases: *aiogram.types.inline\_query\_result.InlineQueryResult*

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.

Note: This will only work in Telegram versions released after 9 April, 2016. Older clients will ignore them.

<https://core.telegram.org/bots/api#inlinequeryresultcachedsticker>

Deserialize object

#### Parameters

- **conf** –
- **kwargs** –

## InlineQueryResultCachedDocument

```

class aiogram.types.inline_query_result.InlineQueryResultCachedDocument(*,
    id:
        String,
    title:
        String,
    document_file_id:
        String,
    description:
        Optional[String]
    =
        None,
    caption:
        Optional[String]
    =
        None,
    parse_mode:
        Optional[String]
    =
        None,
    caption_entities:
        Optional[List[aiogram.types.m
    =
        None,
    reply_markup:
        Optional[aiogram.types.inline
    =
        None,
    input_message_content:
        Optional[aiogram.types.input
    =
        None)

```

Bases: *aiogram.types.inline\_query\_result.InlineQueryResult*

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.

Note: This will only work in Telegram versions released after 9 April, 2016. Older clients will ignore them.

<https://core.telegram.org/bots/api#inlinequeryresultcacheddocument>

Deserialize object

#### Parameters

- **conf** –
- **kwargs** –

### InlineQueryResultCachedVideo

```
class aiogram.types.inline_query_result.InlineQueryResultCachedVideo(*, id:
    String,
    video_file_id:
    String,
    title:
    String,
    de-
    scrip-
    tion:
    Op-
    tional[String]
    =
    None,
    cap-
    tion:
    Op-
    tional[String]
    =
    None,
    parse_mode:
    Op-
    tional[String]
    =
    None,
    cap-
    tion_entities:
    Op-
    tional[List[aiogram.types.messa
    =
    None,
    re-
    ply_markup:
    Op-
    tional[aiogram.types.inline_key
    =
    None,
    in-
    put_message_content:
    Op-
    tional[aiogram.types.input_mes
    =
    None)

Bases: aiogram.types.inline_query_result.InlineQueryResult
```

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.

<https://core.telegram.org/bots/api#inlinequeryresultcachedvideo>

Deserialize object

#### Parameters

- **conf** –
- **kwargs** –

### InlineQueryResultCachedVoice

```
class aiogram.types.inline_query_result.InlineQueryResultCachedVoice(*, id:
    String,
    voice_file_id:
    String,
    title:
    String,
    caption:
    Optional[String]
    =
    None,
    parse_mode:
    Optional[String]
    =
    None,
    caption_entities:
    Optional[List[aiogram.types.messa
    =
    None,
    reply_markup:
    Optional[aiogram.types.inline_key
    =
    None,
    input_message_content:
    Optional[aiogram.types.input_mes
    =
    None)
```

Bases: `aiogram.types.inline_query_result.InlineQueryResult`

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.

Note: This will only work in Telegram versions released after 9 April, 2016. Older clients will ignore them.

<https://core.telegram.org/bots/api#inlinequeryresultcachedvoice>

Deserialize object

#### Parameters

- **conf** –
- **kwargs** –

### InlineQueryResultCachedAudio

```
class aiogram.types.inline_query_result.InlineQueryResultCachedAudio(*, id:
    String,
    audio_file_id:
    String,
    caption:
    Optional[String]
    =
    None,
    parse_mode:
    Optional[String]
    =
    None,
    caption_entities:
    Optional[List[aiogram.types.messages.MessageEntity]]
    =
    None,
    reply_markup:
    Optional[aiogram.types.inline_keyboard.InlineKeyboardMarkup]
    =
    None,
    input_message_content:
    Optional[aiogram.types.input_message_content.InputMessageContent]
    =
    None)
```

Bases: `aiogram.types.inline_query_result.InlineQueryResult`

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.

Note: This will only work in Telegram versions released after 9 April, 2016. Older clients will ignore them.

<https://core.telegram.org/bots/api#inlinequeryresultcachedaudio>

Deserialize object

#### Parameters

- **conf** –
- **kwargs** –

## InputFile

**class** aiogram.types.input\_file.**InputFile** (*path\_or\_bytesio: Union[str, io.IOBase, pathlib.Path], filename=None, conf=None*)

Bases: *aiogram.types.base.TelegramObject*

This object represents the contents of a file to be uploaded. Must be posted using multipart/form-data in the usual way that files are uploaded via the browser.

Also that is not typical TelegramObject!

<https://core.telegram.org/bots/api#inputfile>

#### Parameters

- **path\_or\_bytesio** –
- **filename** –
- **conf** –

**get\_filename**() → str  
Get file name

**Returns** name

**get\_file**()  
Get file object

**Returns**

**classmethod from\_url** (*url, filename=None, chunk\_size=65536*)  
Download file from URL

Manually is not required action. You can send urls instead!

#### Parameters

- **url** – target URL
- **filename** – optional. set custom file name
- **chunk\_size** –

**Returns** InputFile

**save** (*filename, chunk\_size=65536*)  
Write file to disk

#### Parameters

- **filename** –
- **chunk\_size** –

**to\_python()**

Get object as JSON serializable

**Returns**

**classmethod to\_object(data)**

Deserialize object

**Parameters data –**

**Returns**

## PreCheckoutQuery

**class** aiogram.types.pre\_checkout\_query.**PreCheckoutQuery** (*conf: Optional[Dict[str, Any]] = None, \*\*kwargs: Any*)

Bases: *aiogram.types.base.TelegramObject*

This object contains information about an incoming pre-checkout query. Your bot can offer users HTML5 games to play solo or to compete against each other in groups and one-on-one chats.

Create games via @BotFather using the /newgame command.

Please note that this kind of power requires responsibility: you will need to accept the terms for each game that your bots will be offering.

<https://core.telegram.org/bots/api#precheckoutquery>

Deserialize object

**Parameters**

- **conf** –
- **kwargs** –

## Voice

**class** aiogram.types.voice.**Voice** (*conf: Optional[Dict[str, Any]] = None, \*\*kwargs: Any*)

Bases: *aiogram.types.base.TelegramObject, aiogram.types.mixins.Downloadable*

This object represents a voice note.

<https://core.telegram.org/bots/api#voice>

Deserialize object

**Parameters**

- **conf** –
- **kwargs** –



## InputMessageContent

```
class aiogram.types.input_message_content.InputMessageContent (conf: Optional[Dict[str, Any]] = None,
**kwargs: Any)
```

Bases: *aiogram.types.base.TelegramObject*

This object represents the content of a message to be sent as a result of an inline query.

Telegram clients currently support the following 4 types

<https://core.telegram.org/bots/api#inputmessagecontent>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

## InputContactMessageContent

```
class aiogram.types.input_message_content.InputContactMessageContent (phone_number: String,
first_name: Optional[String] = None,
last_name: Optional[String] = None)
```

Bases: *aiogram.types.input\_message\_content.InputMessageContent*

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

Note: This will only work in Telegram versions released after 9 April, 2016. Older clients will ignore them.

<https://core.telegram.org/bots/api#inputcontactmessagecontent>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

## InputLocationMessageContent

```
class aiogram.types.input_message_content.InputLocationMessageContent (latitude:
                                                                    Float,
                                                                    longi-
                                                                    tude:
                                                                    Float,
                                                                    hori-
                                                                    zon-
                                                                    tal_accuracy:
                                                                    Op-
                                                                    tional[Float]
                                                                    =
                                                                    None,
                                                                    live_period:
                                                                    Op-
                                                                    tional[Integer]
                                                                    =
                                                                    None,
                                                                    head-
                                                                    ing:
                                                                    Op-
                                                                    tional[Integer]
                                                                    =
                                                                    None,
                                                                    prox-
                                                                    im-
                                                                    ity_alert_radius:
                                                                    Op-
                                                                    tional[Integer]
                                                                    =
                                                                    None)
```

Bases: *aiogram.types.input\_message\_content.InputMessageContent*

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

<https://core.telegram.org/bots/api#inputlocationmessagecontent>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

## InputTextMessageContent

```

class aiogram.types.input_message_content.InputTextMessageContent (message_text:
                                                                    Op-
                                                                    tional[String]
                                                                    = None,
                                                                    parse_mode:
                                                                    Op-
                                                                    tional[String]
                                                                    = None,
                                                                    cap-
                                                                    tion_entities:
                                                                    Op-
                                                                    tional[List[aiogram.types.message_e
                                                                    = None,
                                                                    dis-
                                                                    able_web_page_preview:
                                                                    Op-
                                                                    tional[Boolean]
                                                                    = None)

```

Bases: *aiogram.types.input\_message\_content.InputMessageContent*

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

<https://core.telegram.org/bots/api#inputtextmessagecontent>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

## InputVenueMessageContent

```
class aiogram.types.input_message_content.InputVenueMessageContent (latitude:
    Optional[Float]
    = None,
    longitude: Optional[Float]
    = None,
    title: Optional[String]
    = None,
    address: Optional[String]
    = None,
    foursquare_id: Optional[String]
    = None,
    foursquare_type: Optional[String]
    = None,
    google_place_id: Optional[String]
    = None,
    google_place_type: Optional[String]
    = None)
```

Bases: *aiogram.types.input\_message\_content.InputMessageContent*

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

Note: This will only work in Telegram versions released after 9 April, 2016. Older clients will ignore them.

<https://core.telegram.org/bots/api#inputvenuemessagecontent>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

## Update

**class** aiogram.types.update.**Update** (*conf: Optional[Dict[str, Any]] = None, \*\*kwargs: Any*)  
 Bases: *aiogram.types.base.TelegramObject*

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

<https://core.telegram.org/bots/api#update>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

## AllowedUpdates

**class** aiogram.types.update.**AllowedUpdates**  
 Bases: *aiogram.utils.helper.Helper*

Helper for allowed\_updates parameter in getUpdates and setWebhook methods.

You can use &, + or | operators for make combination of allowed updates.

### Example:

```
>>> bot.get_updates(allowed_updates=AllowedUpdates.MESSAGE + AllowedUpdates.
↳EDITED_MESSAGE)
```

## PhotoSize

**class** aiogram.types.photo\_size.**PhotoSize** (*conf: Optional[Dict[str, Any]] = None, \*\*kwargs: Any*)  
 Bases: *aiogram.types.base.TelegramObject, aiogram.types.mixins.Downloadable*

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

<https://core.telegram.org/bots/api#photosize>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

## Venue

```
class aiogram.types.venue.Venue (conf: Optional[Dict[str, Any]] = None, **kwargs: Any)
    Bases: aiogram.types.base.TelegramObject
```

This object represents a venue.

<https://core.telegram.org/bots/api#venue>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

## ChosenInlineResult

```
class aiogram.types.chosen_inline_result.ChosenInlineResult (conf: Optional[Dict[str, Any]] = None,
                                                             **kwargs: Any)
    Bases: aiogram.types.base.TelegramObject
```

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

Note: It is necessary to enable inline feedback via @Botfather in order to receive these objects in updates. Your bot can accept payments from Telegram users. Please see the introduction to payments for more details on the process and how to set up payments for your bot. Please note that users will need Telegram v.4.0 or higher to use payments (released on May 18, 2017).

<https://core.telegram.org/bots/api#choseninlineresult>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

## VideoNote

```
class aiogram.types.video_note.VideoNote (conf: Optional[Dict[str, Any]] = None,
                                           **kwargs: Any)
    Bases: aiogram.types.base.TelegramObject, aiogram.types.mixins.Downloadable
```

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

<https://core.telegram.org/bots/api#videonote>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

## WebhookInfo

```
class aiogram.types.webhook_info.WebhookInfo (conf: Optional[Dict[str, Any]] = None,
                                             **kwargs: Any)
```

Bases: *aiogram.types.base.TelegramObject*

Contains information about the current status of a webhook.

<https://core.telegram.org/bots/api#webhookinfo>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

## PassportFile

```
class aiogram.types.passport_file.PassportFile (conf: Optional[Dict[str, Any]] = None,
                                                  **kwargs: Any)
```

Bases: *aiogram.types.base.TelegramObject*

This object represents a file uploaded to Telegram Passport. Currently all Telegram Passport files are in JPEG format when decrypted and don't exceed 10MB.

<https://core.telegram.org/bots/api#passportfile>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

## ChatMember

```
class aiogram.types.chat_member.ChatMember (conf: Optional[Dict[str, Any]] = None,
                                             **kwargs: Any)
```

Bases: *aiogram.types.base.TelegramObject*

This object contains information about one member of a chat.

<https://core.telegram.org/bots/api#chatmember>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

## ChatMemberStatus

```
class aiogram.types.chat_member.ChatMemberStatus
```

Bases: `aiogram.utils.helper.Helper`

Chat member status

## ShippingOption

```
class aiogram.types.shipping_option.ShippingOption(id: String, title: String, prices:  
List[aiogram.types.labeled_price.LabeledPrice]  
= None)
```

Bases: `aiogram.types.base.TelegramObject`

This object represents one shipping option.

<https://core.telegram.org/bots/api#shippingoption>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

**add** (*price: aiogram.types.labeled\_price.LabeledPrice*)

Add price

Parameters **price** –

Returns

## ChatPhoto

```
class aiogram.types.chat_photo.ChatPhoto(conf: Optional[Dict[str, Any]] = None,  
**kwargs: Any)
```

Bases: `aiogram.types.base.TelegramObject`

This object represents a chat photo.

<https://core.telegram.org/bots/api#chatphoto>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

```
async download_small (destination=None, timeout=30, chunk_size=65536, seek=True,  
make_dirs=True)
```

Download file

### Parameters

- **destination** – filename or instance of `io.IOBase`. For e. g. `io.BytesIO`
- **timeout** – Integer
- **chunk\_size** – Integer



- **seek** – Boolean - go to start of file when downloading is finished.
- **make\_dirs** – Make dirs if not exist

**Returns** destination

**async download\_big** (*destination=None, timeout=30, chunk\_size=65536, seek=True, make\_dirs=True*)

Download file

**Parameters**

- **destination** – filename or instance of `io.IOBase`. For e. g. `io.BytesIO`
- **timeout** – Integer
- **chunk\_size** – Integer
- **seek** – Boolean - go to start of file when downloading is finished.
- **make\_dirs** – Make dirs if not exist

**Returns** destination

## Contact

**class** `aiogram.types.contact.Contact` (*conf: Optional[Dict[str, Any]] = None, \*\*kwargs: Any*)

Bases: `aiogram.types.base.TelegramObject`

This object represents a phone contact.

<https://core.telegram.org/bots/api#contact>

Deserialize object

**Parameters**

- **conf** –
- **kwargs** –

## Message

**class** `aiogram.types.message.Message` (*conf: Optional[Dict[str, Any]] = None, \*\*kwargs: Any*)

Bases: `aiogram.types.base.TelegramObject`

This object represents a message.

<https://core.telegram.org/bots/api#message>

Deserialize object

**Parameters**

- **conf** –
- **kwargs** –

**is\_forward**() → bool

Check that the message is forwarded. Only *forward\_date* is required to be in forwarded message.

**Returns** bool

**is\_command**() → bool

Check message text is command

**Returns** bool

**get\_full\_command**() → Optional[Tuple[str, str]]  
Split command and args

**Returns** tuple of (command, args)

**get\_command**(*pure=False*) → Optional[str]  
Get command from message

**Returns**

**get\_args**() → Optional[str]  
Get arguments

**Returns**

**parse\_entities**(*as\_html=True*) → str  
Text or caption formatted as HTML or Markdown.

**Returns** str

**property md\_text**  
Text or caption formatted as markdown.

**Returns** str

**property html\_text**  
Text or caption formatted as HTML

**Returns** str

**property url**  
Get URL for the message

**Returns** str

**link**(*text, as\_html=True*) → str  
Generate URL for using in text messages with HTML or MD parse mode

**Parameters**

- **text** – link label
- **as\_html** – generate as HTML

**Returns** str

**async answer**(*text: String, parse\_mode: Optional[String] = None, entities: Optional[List[aiogram.types.message\_entity.MessageEntity]] = None, disable\_web\_page\_preview: Optional[Boolean] = None, disable\_notification: Optional[Boolean] = None, allow\_sending\_without\_reply: Optional[Boolean] = None, reply\_markup: Optional[Union[aiogram.types.inline\_keyboard.InlineKeyboardMarkup, aiogram.types.reply\_keyboard.ReplyKeyboardMarkup, aiogram.types.reply\_keyboard.ReplyKeyboardRemove, aiogram.types.force\_reply.ForceReply]] = None, reply: Boolean = False*) → *aiogram.types.message.Message*

Answer to this message

**Parameters**

- **text** (*base.String*) – Text of the message to be sent

- **parse\_mode** (`typing.Optional[base.String]`) – Send Markdown or HTML, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in your bot's message.
- **entities** (`typing.Optional[typing.List[MessageEntity]]`) – List of special entities that appear in message text, which can be specified instead of `parse_mode`
- **disable\_web\_page\_preview** (`typing.Optional[base.Boolean]`) – Disables link previews for links in this message
- **disable\_notification** (`typing.Optional[base.Boolean]`) – Sends the message silently. Users will receive a notification with no sound
- **allow\_sending\_without\_reply** (`typing.Optional[base.Boolean]`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- **reply\_markup** (`typing.Union[types.InlineKeyboardMarkup, types.ReplyKeyboardMarkup, types.ReplyKeyboardRemove, types.ForceReply, None]`) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user
- **reply** (`base.Boolean`) – fill 'reply\_to\_message\_id'

**Returns** On success, the sent Message is returned

**Return type** `types.Message`

```
async answer_photo(photo: Union[InputFile, String], caption: Optional[String] =
    None, parse_mode: Optional[String] = None, caption_entities:
    Optional[List[aiogram.types.message_entity.MessageEntity]] =
    None, disable_notification: Optional[Boolean] = None, al-
    low_sending_without_reply: Optional[Boolean] = None, reply_markup:
    Optional[Union[aiogram.types.inline_keyboard.InlineKeyboardMarkup,
    aiogram.types.reply_keyboard.ReplyKeyboardMarkup,
    aiogram.types.reply_keyboard.ReplyKeyboardRemove,
    aiogram.types.force_reply.ForceReply]] = None, reply: Boolean = False)
    → aiogram.types.message.Message
```

Use this method to send photos.

Source: <https://core.telegram.org/bots/api#sendphoto>

#### Parameters

- **photo** (`typing.Union[base.InputFile, base.String]`) – Photo to send
- **caption** (`typing.Optional[base.String]`) – Photo caption (may also be used when resending photos by `file_id`), 0-1024 characters
- **parse\_mode** (`typing.Optional[base.String]`) – Send Markdown or HTML, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in your bot's message.
- **caption\_entities** (`typing.Optional[typing.List[MessageEntity]]`) – List of special entities that appear in message text, which can be specified instead of `parse_mode`
- **disable\_notification** (`typing.Optional[base.Boolean]`) – Sends the message silently. Users will receive a notification with no sound

- **allow\_sending\_without\_reply** (`typing.Optional[base.Boolean]`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- **reply\_markup** (`typing.Union[types.InlineKeyboardMarkup, types.ReplyKeyboardMarkup, types.ReplyKeyboardRemove, types.ForceReply, None]`) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user
- **reply** (`base.Boolean`) – fill 'reply\_to\_message\_id'

**Returns** On success, the sent Message is returned

**Return type** `types.Message`

```
async answer_audio (audio: Union[InputFile, String], caption: Optional[String] = None,
                    parse_mode: Optional[String] = None, caption_entities: Optional[List[aiogram.types.message_entity.MessageEntity]] = None,
                    duration: Optional[Integer] = None, performer: Optional[String] = None, title: Optional[String] = None, thumb: Optional[Union[InputFile, String]] = None,
                    disable_notification: Optional[Boolean] = None, allow_sending_without_reply: Optional[Boolean] = None, reply_markup:
                    Optional[Union[aiogram.types.inline_keyboard.InlineKeyboardMarkup,
                    aiogram.types.reply_keyboard.ReplyKeyboardMarkup,
                    aiogram.types.reply_keyboard.ReplyKeyboardRemove,
                    aiogram.types.force_reply.ForceReply]] = None, reply: Boolean = False)
                    → aiogram.types.message.Message
```

Use this method to send audio files, if you want Telegram clients to display them in the music player. Your audio must be in the .mp3 format.

For sending voice messages, use the `sendVoice` method instead.

Source: <https://core.telegram.org/bots/api#sendaudio>

#### Parameters

- **audio** (`typing.Union[base.InputFile, base.String]`) – Audio file to send.
- **caption** (`typing.Optional[base.String]`) – Audio caption, 0-200 characters
- **parse\_mode** (`typing.Optional[base.String]`) – Send Markdown or HTML, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in your bot's message.
- **caption\_entities** (`typing.Optional[typing.List[MessageEntity]]`) – List of special entities that appear in message text, which can be specified instead of `parse_mode`
- **duration** (`typing.Optional[base.Integer]`) – Duration of the audio in seconds
- **performer** (`typing.Optional[base.String]`) – Performer
- **title** (`typing.Optional[base.String]`) – Track name
- **thumb** (`typing.Union[typing.Union[base.InputFile, base.String], None]`) – Thumbnail of the file sent. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320.

- **disable\_notification** (`typing.Optional[base.Boolean]`) – Sends the message silently. Users will receive a notification with no sound.
- **allow\_sending\_without\_reply** (`typing.Optional[base.Boolean]`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- **reply\_markup** (`typing.Union[types.InlineKeyboardMarkup, types.ReplyKeyboardMarkup, types.ReplyKeyboardRemove, types.ForceReply, None]`) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user
- **reply** (`base.Boolean`) – fill 'reply\_to\_message\_id'

**Returns** On success, the sent Message is returned.

**Return type** `types.Message`

```
async answer_animation (animation: Union[InputFile, String], duration: Optional[Integer]
                        = None, width: Optional[Integer] = None, height: Op-
                        tional[Integer] = None, thumb: Optional[Union[InputFile,
                        String]] = None, caption: Optional[String] = None,
                        parse_mode: Optional[String] = None, caption_entities:
                        Optional[List[aiogram.types.message_entity.MessageEntity]]
                        = None, disable_notification: Optional[Boolean]
                        = None, allow_sending_without_reply: Op-
                        tional[Boolean] = None, reply_markup: Op-
                        tional[Union[aiogram.types.inline_keyboard.InlineKeyboardMarkup,
                        aiogram.types.reply_keyboard.ReplyKeyboardMarkup,
                        aiogram.types.reply_keyboard.ReplyKeyboardRemove,
                        aiogram.types.force_reply.ForceReply]] = None, reply: Boolean
                        = False) → aiogram.types.message.Message
```

Use this method to send animation files (GIF or H.264/MPEG-4 AVC video without sound).

On success, the sent Message is returned. Bots can currently send animation files of up to 50 MB in size, this limit may be changed in the future.

Source <https://core.telegram.org/bots/api#sendanimation>

#### Parameters

- **animation** (`typing.Union[base.InputFile, base.String]`) – Animation to send. Pass a file\_id as String to send an animation that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get an animation from the Internet, or upload a new animation using multipart/form-data
- **duration** (`typing.Optional[base.Integer]`) – Duration of sent animation in seconds
- **width** (`typing.Optional[base.Integer]`) – Animation width
- **height** (`typing.Optional[base.Integer]`) – Animation height
- **thumb** (`typing.Union[typing.Union[base.InputFile, base.String], None]`) – Thumbnail of the file sent. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320.
- **caption** (`typing.Optional[base.String]`) – Animation caption (may also be used when resending animation by file\_id), 0-1024 characters

- **parse\_mode** (`typing.Optional[base.String]`) – Send Markdown or HTML, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in the media caption
- **caption\_entities** (`typing.Optional[typing.List[MessageEntity]]`) – List of special entities that appear in message text, which can be specified instead of `parse_mode`
- **disable\_notification** (`typing.Optional[base.Boolean]`) – Sends the message silently. Users will receive a notification with no sound
- **allow\_sending\_without\_reply** (`typing.Optional[base.Boolean]`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- **reply\_markup** (`typing.Union[typing.Union[types.InlineKeyboardMarkup, types.ReplyKeyboardMarkup, types.ReplyKeyboardRemove, types.ForceReply], None]`) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user
- **reply** (`base.Boolean`) – fill `'reply_to_message_id'`

**Returns** On success, the sent Message is returned

**Return type** `types.Message`

```
async answer_document (document: Union[InputFile, String], thumb: Optional[Union[InputFile, String]] = None, caption: Optional[String] = None, parse_mode: Optional[String] = None, caption_entities: Optional[List[aiogram.types.message_entity.MessageEntity]] = None, disable_content_type_detection: Optional[Boolean] = None, disable_notification: Optional[Boolean] = None, allow_sending_without_reply: Optional[Boolean] = None, reply_markup: Optional[Union[aiogram.types.inline_keyboard.InlineKeyboardMarkup, aiogram.types.reply_keyboard.ReplyKeyboardMarkup, aiogram.types.reply_keyboard.ReplyKeyboardRemove, aiogram.types.force_reply.ForceReply]] = None, reply: Boolean = False) → aiogram.types.message.Message
```

Use this method to send general files. On success, the sent Message is returned. Bots can currently send files of any type of up to 50 MB in size, this limit may be changed in the future.

Source: <https://core.telegram.org/bots/api#senddocument>

#### Parameters

- **document** (`typing.Union[base.InputFile, base.String]`) – File to send
- **thumb** (`typing.Union[base.InputFile, base.String, None]`) – Thumbnail of the file sent
- **caption** (`typing.Optional[base.String]`) – Document caption (may also be used when resending documents by `file_id`), 0-1024 characters
- **disable\_content\_type\_detection** (`typing.Optional[base.Boolean]`) – Disables automatic server-side content type detection for files uploaded using multipart/form-data
- **parse\_mode** (`typing.Optional[base.String]`) – Send Markdown or HTML, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in your bot's message.

- **caption\_entities** (`typing.Optional[typing.List[MessageEntity]]`) – List of special entities that appear in message text, which can be specified instead of `parse_mode`
- **disable\_notification** (`typing.Optional[base.Boolean]`) – Sends the message silently. Users will receive a notification with no sound
- **allow\_sending\_without\_reply** (`typing.Optional[base.Boolean]`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- **reply\_markup** (`typing.Union[types.InlineKeyboardMarkup, types.ReplyKeyboardMarkup, types.ReplyKeyboardRemove, types.ForceReply, None]`) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user
- **reply** (`typing.Optional[base.Boolean]`) – True if the message is a reply

**Returns** On success, the sent Message is returned

**Return type** `types.Message`

```
async answer_video (video: Union[InputFile, String], duration: Optional[Integer] = None,
                    width: Optional[Integer] = None, height: Optional[Integer] = None,
                    thumb: Optional[Union[InputFile, String]] = None, caption: Optional[String] = None,
                    parse_mode: Optional[String] = None, caption_entities: Optional[List[aiogram.types.message_entity.MessageEntity]]
                    = None, supports_streaming: Optional[Boolean] = None,
                    disable_notification: Optional[Boolean] = None,
                    allow_sending_without_reply: Optional[Boolean] = None,
                    reply_markup: Optional[Union[aiogram.types.inline_keyboard.InlineKeyboardMarkup,
                    aiogram.types.reply_keyboard.ReplyKeyboardMarkup,
                    aiogram.types.reply_keyboard.ReplyKeyboardRemove,
                    aiogram.types.force_reply.ForceReply]] = None,
                    reply: Boolean = False)
                    → aiogram.types.message.Message
```

Use this method to send video files, Telegram clients support mp4 videos (other formats may be sent as Document).

Source: <https://core.telegram.org/bots/api#sendvideo>

#### Parameters

- **video** (`typing.Union[base.InputFile, base.String]`) – Video to send.
- **duration** (`typing.Optional[base.Integer]`) – Duration of sent video in seconds
- **width** (`typing.Optional[base.Integer]`) – Video width
- **height** (`typing.Optional[base.Integer]`) – Video height
- **thumb** (`typing.Union[base.InputFile, base.String, None]`) – Thumbnail of the file sent. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320.
- **caption** (`typing.Optional[base.String]`) – Video caption (may also be used when resending videos by `file_id`), 0-200 characters
- **parse\_mode** (`typing.Optional[base.String]`) – Send Markdown or HTML, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in the media caption



- **caption\_entities** (`typing.Optional[typing.List[MessageEntity]]`) – List of special entities that appear in message text, which can be specified instead of `parse_mode`
- **supports\_streaming** (`typing.Optional[base.Boolean]`) – Pass `True`, if the uploaded video is suitable for streaming
- **disable\_notification** (`typing.Optional[base.Boolean]`) – Sends the message silently. Users will receive a notification with no sound.
- **allow\_sending\_without\_reply** (`typing.Optional[base.Boolean]`) – Pass `True`, if the message should be sent even if the specified replied-to message is not found
- **reply\_markup** (`typing.Union[types.InlineKeyboardMarkup, types.ReplyKeyboardMarkup, types.ReplyKeyboardRemove, types.ForceReply, None]`) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user
- **reply** (`base.Boolean`) – fill `'reply_to_message_id'`

**Returns** On success, the sent `Message` is returned.

**Return type** `types.Message`

```
async def answer_voice(voice: Union[InputFile, String], caption: Optional[String] =
    None, parse_mode: Optional[String] = None, caption_entities:
    Optional[List[aiogram.types.message_entity.MessageEntity]] =
    None, duration: Optional[Integer] = None, disable_notification:
    Optional[Boolean] = None, allow_sending_without_reply:
    Optional[Boolean] = None, reply_markup: Op-
    tional[Union[aiogram.types.inline_keyboard.InlineKeyboardMarkup,
    aiogram.types.reply_keyboard.ReplyKeyboardMarkup,
    aiogram.types.reply_keyboard.ReplyKeyboardRemove,
    aiogram.types.force_reply.ForceReply]] = None, reply: Boolean = False)
    → aiogram.types.message.Message
```

Use this method to send audio files, if you want Telegram clients to display the file as a playable voice message.

For this to work, your audio must be in an .ogg file encoded with OPUS (other formats may be sent as Audio or Document).

Source: <https://core.telegram.org/bots/api#sendvoice>

#### Parameters

- **voice** (`typing.Union[base.InputFile, base.String]`) – Audio file to send.
- **caption** (`typing.Optional[base.String]`) – Voice message caption, 0-200 characters
- **parse\_mode** (`typing.Optional[base.String]`) – Send Markdown or HTML, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in the media caption
- **caption\_entities** (`typing.Optional[typing.List[MessageEntity]]`) – List of special entities that appear in message text, which can be specified instead of `parse_mode`



- **duration** (`typing.Optional[base.Integer]`) – Duration of the voice message in seconds
- **disable\_notification** (`typing.Optional[base.Boolean]`) – Sends the message silently. Users will receive a notification with no sound.
- **allow\_sending\_without\_reply** (`typing.Optional[base.Boolean]`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- **reply\_markup** (`typing.Union[types.InlineKeyboardMarkup, types.ReplyKeyboardMarkup, types.ReplyKeyboardRemove, types.ForceReply, None]`) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user
- **reply** (`base.Boolean`) – fill 'reply\_to\_message\_id'

**Returns** On success, the sent Message is returned.

**Return type** `types.Message`

```
async answer_video_note (video_note: Union[InputFile, String], duration: Optional[Integer]
                        = None, length: Optional[Integer] = None, thumb: Optional[Union[InputFile, String]] = None, disable_notification:
                        Optional[Boolean] = None, allow_sending_without_reply:
                        Optional[Boolean] = None, reply_markup: Optional[Union[aiogram.types.inline_keyboard.InlineKeyboardMarkup,
                        aiogram.types.reply_keyboard.ReplyKeyboardMarkup,
                        aiogram.types.reply_keyboard.ReplyKeyboardRemove,
                        aiogram.types.force_reply.ForceReply]] = None, reply: Boolean
                        = False) → aiogram.types.message.Message
```

As of v.4.0, Telegram clients support rounded square mp4 videos of up to 1 minute long. Use this method to send video messages.

Source: <https://core.telegram.org/bots/api#sendvideonote>

#### Parameters

- **video\_note** (`typing.Union[base.InputFile, base.String]`) – Video note to send.
- **duration** (`typing.Optional[base.Integer]`) – Duration of sent video in seconds
- **length** (`typing.Optional[base.Integer]`) – Video width and height
- **thumb** (`typing.Union[typing.Union[base.InputFile, base.String], None]`) – Thumbnail of the file sent. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320.
- **disable\_notification** (`typing.Optional[base.Boolean]`) – Sends the message silently. Users will receive a notification with no sound.
- **allow\_sending\_without\_reply** (`typing.Optional[base.Boolean]`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- **reply\_markup** (`typing.Union[types.InlineKeyboardMarkup, types.ReplyKeyboardMarkup, types.ReplyKeyboardRemove, types.ForceReply, None]`) – Additional interface options. A JSON-serialized

object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user

- **reply** (`base.Boolean`) – fill ‘reply\_to\_message\_id’

**Returns** On success, the sent Message is returned.

**Return type** `types.Message`

```
async answer_media_group (media: Union[aiogram.types.input_media.MediaGroup,  
List], disable_notification: Optional[Boolean] = None, al-  
low_sending_without_reply: Optional[Boolean] = None, reply:  
Boolean = False) → List[aiogram.types.message.Message]
```

Use this method to send a group of photos, videos, documents or audios as an album. Documents and audio files can be only group in an album with messages of the same type. On success, an array of Messages that were sent is returned.

Source: <https://core.telegram.org/bots/api#sendmediagroup>

#### Parameters

- **media** (`typing.Union[types.MediaGroup, typing.List]`) – A JSON-serialized array describing photos and videos to be sent
- **disable\_notification** (`typing.Optional[base.Boolean]`) – Sends the message silently. Users will receive a notification with no sound.
- **allow\_sending\_without\_reply** (`typing.Optional[base.Boolean]`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- **reply** (`base.Boolean`) – fill ‘reply\_to\_message\_id’

**Returns** On success, an array of the sent Messages is returned.

**Return type** `typing.List[types.Message]`

```
async answer_location (latitude: Float, longitude: Float, live_period: Op-  
tional[Integer] = None, disable_notification: Op-  
tional[Boolean] = None, allow_sending_without_reply:  
Optional[Boolean] = None, reply_markup: Op-  
tional[Union[aiogram.types.inline_keyboard.InlineKeyboardMarkup,  
aiogram.types.reply_keyboard.ReplyKeyboardMarkup,  
aiogram.types.reply_keyboard.ReplyKeyboardRemove,  
aiogram.types.force_reply.ForceReply]] = None, reply: Boolean =  
False) → aiogram.types.message.Message
```

Use this method to send point on the map.

Source: <https://core.telegram.org/bots/api#sendlocation>

#### Parameters

- **latitude** (`base.Float`) – Latitude of the location
- **longitude** (`base.Float`) – Longitude of the location
- **live\_period** (`typing.Optional[base.Integer]`) – Period in seconds for which the location will be updated
- **disable\_notification** (`typing.Optional[base.Boolean]`) – Sends the message silently. Users will receive a notification with no sound.

- **allow\_sending\_without\_reply** (`typing.Optional[base.Boolean]`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- **reply\_markup** (`typing.Union[types.InlineKeyboardMarkup, types.ReplyKeyboardMarkup, types.ReplyKeyboardRemove, types.ForceReply, None]`) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user
- **reply** (`base.Boolean`) – fill 'reply\_to\_message\_id'

**Returns** On success, the sent Message is returned.

**Return type** `types.Message`

```
async answer_venue (latitude: Float, longitude: Float, title: String, address: String,
    foursquare_id: Optional[String] = None, foursquare_type: Optional[String]
    = None, google_place_id: Optional[String] = None, google_place_type:
    Optional[String] = None, disable_notification: Optional[Boolean] = None,
    allow_sending_without_reply: Optional[Boolean] = None, reply_markup:
    Optional[Union[aiogram.types.inline_keyboard.InlineKeyboardMarkup,
    aiogram.types.reply_keyboard.ReplyKeyboardMarkup,
    aiogram.types.reply_keyboard.ReplyKeyboardRemove,
    aiogram.types.force_reply.ForceReply]] = None, reply: Boolean = False)
    → aiogram.types.message.Message
```

Use this method to send information about a venue.

Source: <https://core.telegram.org/bots/api#sendvenue>

#### Parameters

- **latitude** (`base.Float`) – Latitude of the venue
- **longitude** (`base.Float`) – Longitude of the venue
- **title** (`base.String`) – Name of the venue
- **address** (`base.String`) – Address of the venue
- **foursquare\_id** (`typing.Optional[base.String]`) – Foursquare identifier of the venue
- **foursquare\_type** (`typing.Optional[base.String]`) – Foursquare type of the venue, if known
- **google\_place\_id** (`typing.Optional[base.String]`) – Google Places identifier of the venue
- **google\_place\_type** (`typing.Optional[base.String]`) – Google Places type of the venue. See supported types: [https://developers.google.com/places/web-service/supported\\_types](https://developers.google.com/places/web-service/supported_types)
- **disable\_notification** (`typing.Optional[base.Boolean]`) – Sends the message silently. Users will receive a notification with no sound
- **allow\_sending\_without\_reply** (`typing.Optional[base.Boolean]`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- **reply\_markup** (`typing.Union[types.InlineKeyboardMarkup, types.ReplyKeyboardMarkup, types.ReplyKeyboardRemove, types.ForceReply, None]`) – Additional interface options. A JSON-serialized

object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user

- **reply** (`base.Boolean`) – fill ‘reply\_to\_message\_id’

**Returns** On success, the sent Message is returned.

**Return type** `types.Message`

```
async answer_contact (phone_number: String, first_name: String, last_name: Optional[String]
                        = None, disable_notification: Optional[Boolean] = None, allow_sending_without_reply: Optional[Boolean] = None, reply_markup:
                        Optional[Union[aiogram.types.inline_keyboard.InlineKeyboardMarkup,
                        aiogram.types.reply_keyboard.ReplyKeyboardMarkup,
                        aiogram.types.reply_keyboard.ReplyKeyboardRemove,
                        aiogram.types.force_reply.ForceReply]] = None, reply: Boolean =
                        False) → aiogram.types.message.Message
```

Use this method to send phone contacts.

Source: <https://core.telegram.org/bots/api#sendcontact>

#### Parameters

- **phone\_number** (`base.String`) – Contact’s phone number
- **first\_name** (`base.String`) – Contact’s first name
- **last\_name** (`typing.Optional[base.String]`) – Contact’s last name
- **disable\_notification** (`typing.Optional[base.Boolean]`) – Sends the message silently. Users will receive a notification with no sound.
- **allow\_sending\_without\_reply** (`typing.Optional[base.Boolean]`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- **reply\_markup** (`typing.Union[types.InlineKeyboardMarkup, types.ReplyKeyboardMarkup, types.ReplyKeyboardRemove, types.ForceReply, None]`) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user
- **reply** (`base.Boolean`) – fill ‘reply\_to\_message\_id’

**Returns** On success, the sent Message is returned.

**Return type** `types.Message`

```
async answer_sticker (sticker: Union[InputFile, String], disable_notification: Op-
                        tional[Boolean] = None, allow_sending_without_reply:
                        Optional[Boolean] = None, reply_markup: Op-
                        tional[Union[aiogram.types.inline_keyboard.InlineKeyboardMarkup,
                        aiogram.types.reply_keyboard.ReplyKeyboardMarkup,
                        aiogram.types.reply_keyboard.ReplyKeyboardRemove,
                        aiogram.types.force_reply.ForceReply]] = None, reply: Boolean =
                        False) → aiogram.types.message.Message
```

Use this method to send .webp stickers.

Source: <https://core.telegram.org/bots/api#sendsticker>

#### Parameters

- **sticker** (`typing.Union[base.InputFile, base.String]`) – Sticker to send.

- **disable\_notification** (`typing.Optional[base.Boolean]`) – Sends the message silently. Users will receive a notification with no sound.
- **allow\_sending\_without\_reply** (`typing.Optional[base.Boolean]`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- **reply\_markup** (`typing.Union[types.InlineKeyboardMarkup, types.ReplyKeyboardMarkup, types.ReplyKeyboardRemove, types.ForceReply, None]`) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user
- **reply** (`base.Boolean`) – fill 'reply\_to\_message\_id'

**Returns** On success, the sent Message is returned.

**Return type** `types.Message`

```
async answer_poll (question: String, options: List[String], is_anonymous: Optional[Boolean] = None, type: Optional[String] = None, allows_multiple_answers: Optional[Boolean] = None, correct_option_id: Optional[Integer] = None, explanation: Optional[String] = None, explanation_parse_mode: Optional[String] = None, explanation_entities: Optional[List[aiogram.types.message_entity.MessageEntity]] = None, open_period: Optional[Integer] = None, close_date: Optional[Union[Integer, datetime.datetime, datetime.timedelta]] = None, is_closed: Optional[Boolean] = None, disable_notification: Optional[Boolean] = None, allow_sending_without_reply: Optional[Boolean] = None, reply_markup: Optional[Union[aiogram.types.inline_keyboard.InlineKeyboardMarkup, aiogram.types.reply_keyboard.ReplyKeyboardMarkup, aiogram.types.reply_keyboard.ReplyKeyboardRemove, aiogram.types.force_reply.ForceReply]] = None, reply: Boolean = False) → aiogram.types.message.Message
```

Use this method to send a native poll. On success, the sent Message is returned.

Source: <https://core.telegram.org/bots/api#sendpoll>

#### Parameters

- **question** (`base.String`) – Poll question, 1-255 characters
- **options** (`typing.List[base.String]`) – List of answer options, 2-10 strings 1-100 characters each
- **is\_anonymous** (`typing.Optional[base.Boolean]`) – True, if the poll needs to be anonymous, defaults to True
- **type** (`typing.Optional[base.String]`) – Poll type, “quiz” or “regular”, defaults to “regular”
- **allows\_multiple\_answers** (`typing.Optional[base.Boolean]`) – True, if the poll allows multiple answers, ignored for polls in quiz mode, defaults to False
- **correct\_option\_id** (`typing.Optional[base.Integer]`) – 0-based identifier of the correct answer option, required for polls in quiz mode
- **explanation** (`typing.Optional[base.String]`) – Text that is shown when a user chooses an incorrect answer or taps on the lamp icon in a quiz-style poll, 0-200 characters with at most 2 line feeds after entities parsing

- **explanation\_parse\_mode** (`typing.Optional[base.String]`) – Mode for parsing entities in the explanation. See formatting options for more details.
- **explanation\_entities** (`typing.Optional[typing.List[MessageEntity]]`) – List of special entities that appear in message text, which can be specified instead of `parse_mode`
- **open\_period** (`typing.Optional[base.Integer]`) – Amount of time in seconds the poll will be active after creation, 5-600. Can't be used together with `close_date`.
- **close\_date** (`typing.Union[base.Integer, datetime.datetime, datetime.timedelta, None]`) – Point in time (Unix timestamp) when the poll will be automatically closed. Must be at least 5 and no more than 600 seconds in the future. Can't be used together with `open_period`.
- **is\_closed** (`typing.Optional[base.Boolean]`) – Pass True, if the poll needs to be immediately closed
- **disable\_notification** (`typing.Optional[base.Boolean]`) – Sends the message silently. Users will receive a notification with no sound.
- **allow\_sending\_without\_reply** (`typing.Optional[base.Boolean]`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- **reply\_markup** (`typing.Union[types.InlineKeyboardMarkup, types.ReplyKeyboardMarkup, types.ReplyKeyboardRemove, types.ForceReply, None]`) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user
- **reply** (`base.Boolean`) – fill 'reply\_to\_message\_id'

**Returns** On success, the sent Message is returned

**Return type** `types.Message`

```
async def answer_dice(emoji: Optional[String] = None, disable_notification: Optional[Boolean] = None, allow_sending_without_reply: Optional[Boolean] = None, reply_markup: Optional[Union[aiogram.types.inline_keyboard.InlineKeyboardMarkup, aiogram.types.reply_keyboard.ReplyKeyboardMarkup, aiogram.types.reply_keyboard.ReplyKeyboardRemove, aiogram.types.force_reply.ForceReply]] = None, reply: Boolean = False) → aiogram.types.message.Message
```

Use this method to send an animated emoji that will display a random value. On success, the sent Message is returned.

Source: <https://core.telegram.org/bots/api#senddice>

#### Parameters

- **emoji** (`typing.Optional[base.String]`) – Emoji on which the dice throw animation is based. Currently, must be one of “”, “”, “”, “”, or “”. Dice can have values 1-6 for “” and “”, values 1-5 for “” and “”, and values 1-64 for “”. Defaults to “”
- **disable\_notification** (`typing.Optional[base.Boolean]`) – Sends the message silently. Users will receive a notification with no sound
- **allow\_sending\_without\_reply** (`typing.Optional[base.Boolean]`) – Pass True, if the message should be sent even if the specified replied-to message is not found

- **reply\_markup** (`typing.Union[types.InlineKeyboardMarkup, types.ReplyKeyboardMarkup, types.ReplyKeyboardRemove, types.ForceReply, None]`) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user
- **reply** (`base.Boolean`) – fill ‘reply\_to\_message\_id’

**Returns** On success, the sent Message is returned.

**Return type** `types.Message`

```
async reply (text: String, parse_mode: Optional[String] = None, entities: Optional[List[aiogram.types.message_entity.MessageEntity]] = None, disable_web_page_preview: Optional[Boolean] = None, disable_notification: Optional[Boolean] = None, allow_sending_without_reply: Optional[Boolean] = None, reply_markup: Optional[Union[aiogram.types.inline_keyboard.InlineKeyboardMarkup, aiogram.types.reply_keyboard.ReplyKeyboardMarkup, aiogram.types.reply_keyboard.ReplyKeyboardRemove, aiogram.types.force_reply.ForceReply]] = None, reply: Boolean = True) → aiogram.types.message.Message
```

Reply to this message

**Parameters**

- **text** (`base.String`) – Text of the message to be sent
- **parse\_mode** (`typing.Optional[base.String]`) – Send Markdown or HTML, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in your bot’s message.
- **entities** (`typing.Optional[typing.List[MessageEntity]]`) – List of special entities that appear in message text, which can be specified instead of parse\_mode
- **disable\_web\_page\_preview** (`typing.Optional[base.Boolean]`) – Disables link previews for links in this message
- **disable\_notification** (`typing.Optional[base.Boolean]`) – Sends the message silently. Users will receive a notification with no sound
- **allow\_sending\_without\_reply** (`typing.Optional[base.Boolean]`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- **reply\_markup** (`typing.Union[types.InlineKeyboardMarkup, types.ReplyKeyboardMarkup, types.ReplyKeyboardRemove, types.ForceReply, None]`) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user
- **reply** (`base.Boolean`) – fill ‘reply\_to\_message\_id’

**Returns** On success, the sent Message is returned

**Return type** `types.Message`



```
async reply_photo (photo: Union[InputFile, String], caption: Optional[String] =  
None, parse_mode: Optional[String] = None, caption_entities:  
Optional[List[aiogram.types.message_entity.MessageEntity]] =  
None, disable_notification: Optional[Boolean] = None, al-  
low_sending_without_reply: Optional[Boolean] = None, reply_markup:  
Optional[Union[aiogram.types.inline_keyboard.InlineKeyboardMarkup,  
aiogram.types.reply_keyboard.ReplyKeyboardMarkup,  
aiogram.types.reply_keyboard.ReplyKeyboardRemove,  
aiogram.types.force_reply.ForceReply]] = None, reply: Boolean = True) →  
aiogram.types.message.Message
```

Use this method to send photos.

Source: <https://core.telegram.org/bots/api#sendphoto>

#### Parameters

- **photo** (typing.Union[base.InputFile, base.String]) – Photo to send
- **caption** (typing.Optional[base.String]) – Photo caption (may also be used when resending photos by file\_id), 0-1024 characters
- **parse\_mode** (typing.Optional[base.String]) – Send Markdown or HTML, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in your bot's message.
- **caption\_entities** (typing.Optional[typing.List[MessageEntity]]) – List of special entities that appear in message text, which can be specified instead of parse\_mode
- **disable\_notification** (typing.Optional[base.Boolean]) – Sends the message silently. Users will receive a notification with no sound
- **allow\_sending\_without\_reply** (typing.Optional[base.Boolean]) – Pass True, if the message should be sent even if the specified replied-to message is not found
- **reply\_markup** (typing.Union[types.InlineKeyboardMarkup, types.ReplyKeyboardMarkup, types.ReplyKeyboardRemove, types.ForceReply, None]) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user
- **reply** (base.Boolean) – fill 'reply\_to\_message\_id'

**Returns** On success, the sent Message is returned

**Return type** types.Message

```
async reply_audio (audio: Union[InputFile, String], caption: Optional[String] =  
None, parse_mode: Optional[String] = None, caption_entities: Op-  
tional[List[aiogram.types.message_entity.MessageEntity]] = None, dura-  
tion: Optional[Integer] = None, performer: Optional[String] = None,  
title: Optional[String] = None, thumb: Optional[Union[InputFile,  
String]] = None, disable_notification: Optional[Boolean] = None, al-  
low_sending_without_reply: Optional[Boolean] = None, reply_markup:  
Optional[Union[aiogram.types.inline_keyboard.InlineKeyboardMarkup,  
aiogram.types.reply_keyboard.ReplyKeyboardMarkup,  
aiogram.types.reply_keyboard.ReplyKeyboardRemove,  
aiogram.types.force_reply.ForceReply]] = None, reply: Boolean = True) →  
aiogram.types.message.Message
```

Use this method to send audio files, if you want Telegram clients to display them in the music player. Your



audio must be in the .mp3 format.

For sending voice messages, use the `sendVoice` method instead.

Source: <https://core.telegram.org/bots/api#sendaudio>

#### Parameters

- **audio** (`typing.Union[base.InputFile, base.String]`) – Audio file to send.
- **caption** (`typing.Optional[base.String]`) – Audio caption, 0-200 characters
- **parse\_mode** (`typing.Optional[base.String]`) – Send Markdown or HTML, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in your bot's message.
- **caption\_entities** (`typing.Optional[typing.List[MessageEntity]]`) – List of special entities that appear in message text, which can be specified instead of `parse_mode`
- **duration** (`typing.Optional[base.Integer]`) – Duration of the audio in seconds
- **performer** (`typing.Optional[base.String]`) – Performer
- **title** (`typing.Optional[base.String]`) – Track name
- **thumb** (`typing.Union[typing.Union[base.InputFile, base.String], None]`) – Thumbnail of the file sent. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320.
- **disable\_notification** (`typing.Optional[base.Boolean]`) – Sends the message silently. Users will receive a notification with no sound.
- **allow\_sending\_without\_reply** (`typing.Optional[base.Boolean]`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- **reply\_markup** (`typing.Union[types.InlineKeyboardMarkup, types.ReplyKeyboardMarkup, types.ReplyKeyboardRemove, types.ForceReply, None]`) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user
- **reply** (`base.Boolean`) – fill 'reply\_to\_message\_id'

**Returns** On success, the sent `Message` is returned.

**Return type** `types.Message`

```
async reply_animation(animation: Union[InputFile, String], duration: Optional[Integer]
                        = None, width: Optional[Integer] = None, height: Op-
                        tional[Integer] = None, thumb: Optional[Union[InputFile,
                        String]] = None, caption: Optional[String] = None,
                        parse_mode: Optional[String] = None, caption_entities:
                        Optional[List[aiogram.types.message_entity.MessageEntity]]
                        = None, disable_notification: Optional[Boolean]
                        = None, allow_sending_without_reply: Op-
                        tional[Boolean] = None, reply_markup: Op-
                        tional[Union[aiogram.types.inline_keyboard.InlineKeyboardMarkup,
                        aiogram.types.reply_keyboard.ReplyKeyboardMarkup,
                        aiogram.types.reply_keyboard.ReplyKeyboardRemove,
                        aiogram.types.force_reply.ForceReply]] = None, reply: Boolean =
                        True) → aiogram.types.message.Message
```

Use this method to send animation files (GIF or H.264/MPEG-4 AVC video without sound).

On success, the sent Message is returned. Bots can currently send animation files of up to 50 MB in size, this limit may be changed in the future.

Source <https://core.telegram.org/bots/api#sendanimation>

#### Parameters

- **animation** (typing.Union[base.InputFile, base.String]) – Animation to send. Pass a file\_id as String to send an animation that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get an animation from the Internet, or upload a new animation using multipart/form-data
- **duration** (typing.Optional[base.Integer]) – Duration of sent animation in seconds
- **width** (typing.Optional[base.Integer]) – Animation width
- **height** (typing.Optional[base.Integer]) – Animation height
- **thumb** (typing.Union[typing.Union[base.InputFile, base.String], None]) – Thumbnail of the file sent. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320.
- **caption** (typing.Optional[base.String]) – Animation caption (may also be used when resending animation by file\_id), 0-1024 characters
- **parse\_mode** (typing.Optional[base.String]) – Send Markdown or HTML, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in the media caption
- **caption\_entities** (typing.Optional[typing.List[MessageEntity]]) – List of special entities that appear in message text, which can be specified instead of parse\_mode
- **disable\_notification** (typing.Optional[base.Boolean]) – Sends the message silently. Users will receive a notification with no sound
- **allow\_sending\_without\_reply** (typing.Optional[base.Boolean]) – Pass True, if the message should be sent even if the specified replied-to message is not found
- **reply\_markup** (typing.Union[typing.Union[types.InlineKeyboardMarkup, types.ReplyKeyboardMarkup, types.ReplyKeyboardRemove, types.ForceReply], None]) – Additional inter-

face options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user

- **reply** (`base.Boolean`) – fill 'reply\_to\_message\_id'

**Returns** On success, the sent Message is returned

**Return type** `types.Message`

```
async reply_document (document: Union[InputFile, String], thumb: Optional[Union[InputFile, String]] = None, caption: Optional[String] = None, parse_mode: Optional[String] = None, caption_entities: Optional[List[aiogram.types.message_entity.MessageEntity]] = None, disable_content_type_detection: Optional[Boolean] = None, disable_notification: Optional[Boolean] = None, allow_sending_without_reply: Optional[Boolean] = None, reply_markup: Optional[Union[aiogram.types.inline_keyboard.InlineKeyboardMarkup, aiogram.types.reply_keyboard.ReplyKeyboardMarkup, aiogram.types.reply_keyboard.ReplyKeyboardRemove, aiogram.types.force_reply.ForceReply]] = None, reply: Boolean = True) → aiogram.types.message.Message
```

Use this method to send general files. On success, the sent Message is returned. Bots can currently send files of any type of up to 50 MB in size, this limit may be changed in the future.

Source: <https://core.telegram.org/bots/api#senddocument>

#### Parameters

- **document** (`typing.Union[base.InputFile, base.String]`) – File to send
- **thumb** (`typing.Union[base.InputFile, base.String, None]`) – Thumbnail of the file sent
- **caption** (`typing.Optional[base.String]`) – Document caption (may also be used when resending documents by file\_id), 0-1024 characters
- **disable\_content\_type\_detection** (`typing.Optional[base.Boolean]`) – Disables automatic server-side content type detection for files uploaded using multipart/form-data
- **parse\_mode** (`typing.Optional[base.String]`) – Send Markdown or HTML, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in your bot's message.
- **caption\_entities** (`typing.Optional[typing.List[MessageEntity]]`) – List of special entities that appear in message text, which can be specified instead of parse\_mode
- **disable\_notification** (`typing.Optional[base.Boolean]`) – Sends the message silently. Users will receive a notification with no sound
- **allow\_sending\_without\_reply** (`typing.Optional[base.Boolean]`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- **reply\_markup** (`typing.Union[types.InlineKeyboardMarkup, types.ReplyKeyboardMarkup, types.ReplyKeyboardRemove, types.ForceReply, None]`) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user
- **reply** (`typing.Optional[base.Boolean]`) – True if the message is a reply

**Returns** On success, the sent Message is returned

**Return type** `types.Message`

```
async reply_video (video: Union[InputFile, String], duration: Optional[Integer] = None,
                    width: Optional[Integer] = None, height: Optional[Integer] = None,
                    thumb: Optional[Union[InputFile, String]] = None, caption: Optional[String] = None,
                    parse_mode: Optional[String] = None, caption_entities: Optional[List[aiogram.types.message_entity.MessageEntity]]
                    = None, supports_streaming: Optional[Boolean] = None,
                    disable_notification: Optional[Boolean] = None,
                    allow_sending_without_reply: Optional[Boolean] = None,
                    reply_markup: Optional[Union[aiogram.types.inline_keyboard.InlineKeyboardMarkup,
                    aiogram.types.reply_keyboard.ReplyKeyboardMarkup,
                    aiogram.types.reply_keyboard.ReplyKeyboardRemove,
                    aiogram.types.force_reply.ForceReply]] = None,
                    reply: Boolean = True) →
                    aiogram.types.message.Message
```

Use this method to send video files, Telegram clients support mp4 videos (other formats may be sent as Document).

Source: <https://core.telegram.org/bots/api#sendvideo>

#### Parameters

- **video** (`typing.Union[base.InputFile, base.String]`) – Video to send.
- **duration** (`typing.Optional[base.Integer]`) – Duration of sent video in seconds
- **width** (`typing.Optional[base.Integer]`) – Video width
- **height** (`typing.Optional[base.Integer]`) – Video height
- **thumb** (`typing.Union[base.InputFile, base.String, None]`) – Thumbnail of the file sent. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320.
- **caption** (`typing.Optional[base.String]`) – Video caption (may also be used when resending videos by `file_id`), 0-200 characters
- **parse\_mode** (`typing.Optional[base.String]`) – Send Markdown or HTML, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in the media caption
- **caption\_entities** (`typing.Optional[typing.List[MessageEntity]]`) – List of special entities that appear in message text, which can be specified instead of `parse_mode`
- **supports\_streaming** (`typing.Optional[base.Boolean]`) – Pass True, if the uploaded video is suitable for streaming
- **disable\_notification** (`typing.Optional[base.Boolean]`) – Sends the message silently. Users will receive a notification with no sound.
- **allow\_sending\_without\_reply** (`typing.Optional[base.Boolean]`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- **reply\_markup** (`typing.Union[types.InlineKeyboardMarkup, types.ReplyKeyboardMarkup, types.ReplyKeyboardRemove, types.ForceReply, None]`) – Additional interface options. A JSON-serialized

object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user

- **reply** (`base.Boolean`) – fill ‘reply\_to\_message\_id’

**Returns** On success, the sent Message is returned.

**Return type** `types.Message`

```
async reply_voice (voice: Union[InputFile, String], caption: Optional[String] = None,
                    parse_mode: Optional[String] = None, caption_entities: Optional[List[aiogram.types.message_entity.MessageEntity]] = None, duration:
                    Optional[Integer] = None, disable_notification: Optional[Boolean] = None, allow_sending_without_reply: Optional[Boolean] = None, reply_markup:
                    Optional[Union[aiogram.types.inline_keyboard.InlineKeyboardMarkup,
                    aiogram.types.reply_keyboard.ReplyKeyboardMarkup,
                    aiogram.types.reply_keyboard.ReplyKeyboardRemove,
                    aiogram.types.force_reply.ForceReply]] = None, reply: Boolean = True) →
                    aiogram.types.message.Message
```

Use this method to send audio files, if you want Telegram clients to display the file as a playable voice message.

For this to work, your audio must be in an .ogg file encoded with OPUS (other formats may be sent as Audio or Document).

Source: <https://core.telegram.org/bots/api#sendvoice>

#### Parameters

- **voice** (`typing.Union[base.InputFile, base.String]`) – Audio file to send.
- **caption** (`typing.Optional[base.String]`) – Voice message caption, 0-200 characters
- **parse\_mode** (`typing.Optional[base.String]`) – Send Markdown or HTML, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in the media caption
- **caption\_entities** (`typing.Optional[typing.List[MessageEntity]]`) – List of special entities that appear in message text, which can be specified instead of parse\_mode
- **duration** (`typing.Optional[base.Integer]`) – Duration of the voice message in seconds
- **disable\_notification** (`typing.Optional[base.Boolean]`) – Sends the message silently. Users will receive a notification with no sound.
- **allow\_sending\_without\_reply** (`typing.Optional[base.Boolean]`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- **reply\_markup** (`typing.Union[types.InlineKeyboardMarkup, types.ReplyKeyboardMarkup, types.ReplyKeyboardRemove, types.ForceReply, None]`) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user
- **reply** (`base.Boolean`) – fill ‘reply\_to\_message\_id’

**Returns** On success, the sent Message is returned.

**Return type** `types.Message`

```
async reply_video_note (video_note: Union[InputFile, String], duration: Optional[Integer]
    = None, length: Optional[Integer] = None, thumb: Op-
    tional[Union[InputFile, String]] = None, disable_notification:
    Optional[Boolean] = None, allow_sending_without_reply:
    Optional[Boolean] = None, reply_markup: Op-
    tional[Union[aiogram.types.inline_keyboard.InlineKeyboardMarkup,
    aiogram.types.reply_keyboard.ReplyKeyboardMarkup,
    aiogram.types.reply_keyboard.ReplyKeyboardRemove,
    aiogram.types.force_reply.ForceReply]] = None, reply: Boolean
    = True) → aiogram.types.message.Message
```

As of v.4.0, Telegram clients support rounded square mp4 videos of up to 1 minute long. Use this method to send video messages.

Source: <https://core.telegram.org/bots/api#sendvideonote>

#### Parameters

- **video\_note** (`typing.Union[base.InputFile, base.String]`) – Video note to send.
- **duration** (`typing.Optional[base.Integer]`) – Duration of sent video in seconds
- **length** (`typing.Optional[base.Integer]`) – Video width and height
- **thumb** (`typing.Union[typing.Union[base.InputFile, base.String], None]`) – Thumbnail of the file sent. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320.
- **disable\_notification** (`typing.Optional[base.Boolean]`) – Sends the message silently. Users will receive a notification with no sound.
- **allow\_sending\_without\_reply** (`typing.Optional[base.Boolean]`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- **reply\_markup** (`:obj: typing.Union[types.InlineKeyboardMarkup, types.ReplyKeyboardMarkup, types.ReplyKeyboardRemove, types.ForceReply, None]`) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user
- **reply** (`base.Boolean`) – fill 'reply\_to\_message\_id'

**Returns** On success, the sent `Message` is returned.

**Return type** `types.Message`

```
async reply_media_group (media: Union[aiogram.types.input_media.MediaGroup,
    List], disable_notification: Optional[Boolean] = None, al-
    low_sending_without_reply: Optional[Boolean] = None, reply:
    Boolean = True) → List[aiogram.types.message.Message]
```

Use this method to send a group of photos, videos, documents or audios as an album. Documents and audio files can be only group in an album with messages of the same type. On success, an array of `Messages` that were sent is returned.

Source: <https://core.telegram.org/bots/api#sendmediagroup>

#### Parameters

- **media** (`typing.Union[types.MediaGroup, typing.List]`) – A JSON-serialized array describing photos and videos to be sent
- **disable\_notification** (`typing.Optional[base.Boolean]`) – Sends the message silently. Users will receive a notification with no sound.
- **allow\_sending\_without\_reply** (`typing.Optional[base.Boolean]`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- **reply** (`base.Boolean`) – fill 'reply\_to\_message\_id'

**Returns** On success, an array of the sent Messages is returned.

**Return type** `typing.List[types.Message]`

**async reply\_location** (*latitude: Float, longitude: Float, live\_period: Optional[Integer] = None, disable\_notification: Optional[Boolean] = None, reply\_markup: Optional[Union[aiogram.types.inline\_keyboard.InlineKeyboardMarkup, aiogram.types.reply\_keyboard.ReplyKeyboardMarkup, aiogram.types.reply\_keyboard.ReplyKeyboardRemove, aiogram.types.force\_reply.ForceReply]] = None, reply: Boolean = True*) → *aiogram.types.message.Message*

Use this method to send point on the map.

Source: <https://core.telegram.org/bots/api#sendlocation>

#### Parameters

- **latitude** (`base.Float`) – Latitude of the location
- **longitude** (`base.Float`) – Longitude of the location
- **live\_period** (`typing.Optional[base.Integer]`) – Period in seconds for which the location will be updated
- **disable\_notification** (`typing.Optional[base.Boolean]`) – Sends the message silently. Users will receive a notification with no sound.
- **reply\_markup** (`typing.Union[types.InlineKeyboardMarkup, types.ReplyKeyboardMarkup, types.ReplyKeyboardRemove, types.ForceReply, None]`) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user
- **reply** (`base.Boolean`) – fill 'reply\_to\_message\_id'

**Returns** On success, the sent Message is returned.

**Return type** `types.Message`

**async reply\_venue** (*latitude: Float, longitude: Float, title: String, address: String, foursquare\_id: Optional[String] = None, foursquare\_type: Optional[String] = None, google\_place\_id: Optional[String] = None, google\_place\_type: Optional[String] = None, disable\_notification: Optional[Boolean] = None, allow\_sending\_without\_reply: Optional[Boolean] = None, reply\_markup: Optional[Union[aiogram.types.inline\_keyboard.InlineKeyboardMarkup, aiogram.types.reply\_keyboard.ReplyKeyboardMarkup, aiogram.types.reply\_keyboard.ReplyKeyboardRemove, aiogram.types.force\_reply.ForceReply]] = None, reply: Boolean = True*) → *aiogram.types.message.Message*

Use this method to send information about a venue.



Source: <https://core.telegram.org/bots/api#sendvenue>

#### Parameters

- **latitude** (`base.Float`) – Latitude of the venue
- **longitude** (`base.Float`) – Longitude of the venue
- **title** (`base.String`) – Name of the venue
- **address** (`base.String`) – Address of the venue
- **foursquare\_id** (`typing.Optional[base.String]`) – Foursquare identifier of the venue
- **foursquare\_type** (`typing.Optional[base.String]`) – Foursquare type of the venue, if known
- **google\_place\_id** (`typing.Optional[base.String]`) – Google Places identifier of the venue
- **google\_place\_type** (`typing.Optional[base.String]`) – Google Places type of the venue. See supported types: [https://developers.google.com/places/web-service/supported\\_types](https://developers.google.com/places/web-service/supported_types)
- **disable\_notification** (`typing.Optional[base.Boolean]`) – Sends the message silently. Users will receive a notification with no sound
- **allow\_sending\_without\_reply** (`typing.Optional[base.Boolean]`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- **reply\_markup** (`typing.Union[types.InlineKeyboardMarkup, types.ReplyKeyboardMarkup, types.ReplyKeyboardRemove, types.ForceReply, None]`) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user
- **reply** (`base.Boolean`) – fill 'reply\_to\_message\_id'

**Returns** On success, the sent Message is returned.

**Return type** `types.Message`

```
async reply_contact (phone_number: String, first_name: String, last_name: Optional[String]  
                     = None, disable_notification: Optional[Boolean] = None, al-  
                     low_sending_without_reply: Optional[Boolean] = None, reply_markup:  
                     Optional[Union[aiogram.types.inline_keyboard.InlineKeyboardMarkup,  
                     aiogram.types.reply_keyboard.ReplyKeyboardMarkup,  
                     aiogram.types.reply_keyboard.ReplyKeyboardRemove,  
                     aiogram.types.force_reply.ForceReply]] = None, reply: Boolean =  
                     True) → aiogram.types.message.Message
```

Use this method to send phone contacts.

Source: <https://core.telegram.org/bots/api#sendcontact>

#### Parameters

- **phone\_number** (`base.String`) – Contact's phone number
- **first\_name** (`base.String`) – Contact's first name
- **last\_name** (`typing.Optional[base.String]`) – Contact's last name



- **disable\_notification** (`typing.Optional[base.Boolean]`) – Sends the message silently. Users will receive a notification with no sound.
- **allow\_sending\_without\_reply** (`typing.Optional[base.Boolean]`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- **reply\_markup** (`typing.Union[types.InlineKeyboardMarkup, types.ReplyKeyboardMarkup, types.ReplyKeyboardRemove, types.ForceReply, None]`) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user
- **reply** (`base.Boolean`) – fill 'reply\_to\_message\_id'

**Returns** On success, the sent Message is returned.

**Return type** `types.Message`

```
async reply_poll (question: String, options: List[String], is_anonymous: Optional[Boolean] = None, type: Optional[String] = None, allows_multiple_answers: Optional[Boolean] = None, correct_option_id: Optional[Integer] = None, explanation: Optional[String] = None, explanation_parse_mode: Optional[String] = None, explanation_entities: Optional[List[aiogram.types.message_entity.MessageEntity]] = None, open_period: Optional[Integer] = None, close_date: Optional[Union[Integer, datetime.datetime, datetime.timedelta]] = None, is_closed: Optional[Boolean] = None, disable_notification: Optional[Boolean] = None, allow_sending_without_reply: Optional[Boolean] = None, reply_markup: Optional[Union[aiogram.types.inline_keyboard.InlineKeyboardMarkup, aiogram.types.reply_keyboard.ReplyKeyboardMarkup, aiogram.types.reply_keyboard.ReplyKeyboardRemove, aiogram.types.force_reply.ForceReply]] = None, reply: Boolean = True)
    → aiogram.types.message.Message
```

Use this method to send a native poll. On success, the sent Message is returned.

Source: <https://core.telegram.org/bots/api#sendpoll>

#### Parameters

- **question** (`base.String`) – Poll question, 1-255 characters
- **options** (`typing.List[base.String]`) – List of answer options, 2-10 strings 1-100 characters each
- **is\_anonymous** (`typing.Optional[base.Boolean]`) – True, if the poll needs to be anonymous, defaults to True
- **type** (`typing.Optional[base.String]`) – Poll type, “quiz” or “regular”, defaults to “regular”
- **allows\_multiple\_answers** (`typing.Optional[base.Boolean]`) – True, if the poll allows multiple answers, ignored for polls in quiz mode, defaults to False
- **correct\_option\_id** (`typing.Optional[base.Integer]`) – 0-based identifier of the correct answer option, required for polls in quiz mode
- **explanation** (`typing.Optional[base.String]`) – Text that is shown when a user chooses an incorrect answer or taps on the lamp icon in a quiz-style poll, 0-200 characters with at most 2 line feeds after entities parsing

- **explanation\_parse\_mode** (`typing.Optional[base.String]`) – Mode for parsing entities in the explanation. See formatting options for more details.
- **explanation\_entities** (`typing.Optional[typing.List[MessageEntity]]`) – List of special entities that appear in message text, which can be specified instead of `parse_mode`
- **open\_period** (`typing.Optional[base.Integer]`) – Amount of time in seconds the poll will be active after creation, 5-600. Can't be used together with `close_date`.
- **close\_date** (`typing.Union[base.Integer, datetime.datetime, datetime.timedelta, None]`) – Point in time (Unix timestamp) when the poll will be automatically closed. Must be at least 5 and no more than 600 seconds in the future. Can't be used together with `open_period`.
- **is\_closed** (`typing.Optional[base.Boolean]`) – Pass True, if the poll needs to be immediately closed
- **disable\_notification** (`typing.Optional[base.Boolean]`) – Sends the message silently. Users will receive a notification with no sound.
- **allow\_sending\_without\_reply** (`typing.Optional[base.Boolean]`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- **reply\_markup** (`typing.Union[types.InlineKeyboardMarkup, types.ReplyKeyboardMarkup, types.ReplyKeyboardRemove, types.ForceReply, None]`) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user
- **reply** (`base.Boolean`) – fill 'reply\_to\_message\_id'

**Returns** On success, the sent Message is returned

**Return type** `types.Message`

```
async def reply_sticker(sticker: Union[InputFile, String], disable_notification: Optional[Boolean] = None, allow_sending_without_reply: Optional[Boolean] = None, reply_markup: Optional[Union[aiogram.types.inline_keyboard.InlineKeyboardMarkup, aiogram.types.reply_keyboard.ReplyKeyboardMarkup, aiogram.types.reply_keyboard.ReplyKeyboardRemove, aiogram.types.force_reply.ForceReply]] = None, reply: Boolean = True) → aiogram.types.message.Message
```

Use this method to send .webp stickers.

Source: <https://core.telegram.org/bots/api#sendsticker>

#### Parameters

- **sticker** (`typing.Union[base.InputFile, base.String]`) – Sticker to send.
- **disable\_notification** (`typing.Optional[base.Boolean]`) – Sends the message silently. Users will receive a notification with no sound.
- **allow\_sending\_without\_reply** (`typing.Optional[base.Boolean]`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- **reply\_markup** (`typing.Union[types.InlineKeyboardMarkup, types.ReplyKeyboardMarkup, types.ReplyKeyboardRemove,`

`types.ForceReply, None]`) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user

- **reply** (`base.Boolean`) – fill ‘reply\_to\_message\_id’

**Returns** On success, the sent Message is returned.

**Return type** `types.Message`

```
async reply_dice (emoji: Optional[String] = None, disable_notification: Optional[Boolean] = None, allow_sending_without_reply: Optional[Boolean] = None, reply_markup: Optional[Union[aiogram.types.inline_keyboard.InlineKeyboardMarkup, aiogram.types.reply_keyboard.ReplyKeyboardMarkup, aiogram.types.reply_keyboard.ReplyKeyboardRemove, aiogram.types.force_reply.ForceReply]] = None, reply: Boolean = True)
    → aiogram.types.message.Message
```

Use this method to send an animated emoji that will display a random value. On success, the sent Message is returned.

Source: <https://core.telegram.org/bots/api#senddice>

#### Parameters

- **emoji** (`typing.Optional[base.String]`) – Emoji on which the dice throw animation is based. Currently, must be one of “”, “”, “”, “”, or “”. Dice can have values 1-6 for “” and “”, values 1-5 for “” and “”, and values 1-64 for “”. Defaults to “”
- **disable\_notification** (`typing.Optional[base.Boolean]`) – Sends the message silently. Users will receive a notification with no sound
- **allow\_sending\_without\_reply** (`typing.Optional[base.Boolean]`) – Pass True, if the message should be sent even if the specified replied-to message is not found
- **reply\_markup** (`typing.Union[types.InlineKeyboardMarkup, types.ReplyKeyboardMarkup, types.ReplyKeyboardRemove, types.ForceReply, None]`) – Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user
- **reply** (`base.Boolean`) – fill ‘reply\_to\_message\_id’

**Returns** On success, the sent Message is returned.

**Return type** `types.Message`

```
async forward (chat_id: Union[Integer, String], disable_notification: Optional[Boolean] = None)
    → aiogram.types.message.Message
```

Forward this message

Source: <https://core.telegram.org/bots/api#forwardmessage>

#### Parameters

- **chat\_id** (`typing.Union[base.Integer, base.String]`) – Unique identifier for the target chat or username of the target channel
- **disable\_notification** (`typing.Optional[base.Boolean]`) – Sends the message silently. Users will receive a notification with no sound

**Returns** On success, the sent Message is returned

**Return type** `types.Message`

```
async edit_text (text: String, parse_mode: Optional[String] = None, entities: Optional[List[aiogram.types.message_entity.MessageEntity]] = None, disable_web_page_preview: Optional[Boolean] = None, reply_markup: Optional[aiogram.types.inline_keyboard.InlineKeyboardMarkup] = None) → Union[aiogram.types.message.Message, Boolean]
```

Use this method to edit text and game messages sent by the bot or via the bot (for inline bots).

Source: <https://core.telegram.org/bots/api#editmessagetext>

#### Parameters

- **text** (`base.String`) – New text of the message
- **parse\_mode** (`typing.Optional[base.String]`) – Send Markdown or HTML, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in your bot's message.
- **entities** (`typing.Optional[typing.List[MessageEntity]]`) – List of special entities that appear in message text, which can be specified instead of `parse_mode`
- **disable\_web\_page\_preview** (`typing.Optional[base.Boolean]`) – Disables link previews for links in this message
- **reply\_markup** (`typing.Optional[types.InlineKeyboardMarkup]`) – A JSON-serialized object for an inline keyboard.

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

**Return type** `typing.Union[types.Message, base.Boolean]`

```
async edit_caption (caption: String, parse_mode: Optional[String] = None, caption_entities: Optional[List[aiogram.types.message_entity.MessageEntity]] = None, reply_markup: Optional[aiogram.types.inline_keyboard.InlineKeyboardMarkup] = None) → Union[aiogram.types.message.Message, Boolean]
```

Use this method to edit captions of messages sent by the bot or via the bot (for inline bots).

Source: <https://core.telegram.org/bots/api#editmessagecaption>

#### Parameters

- **caption** (`typing.Optional[base.String]`) – New caption of the message
- **parse\_mode** (`typing.Optional[base.String]`) – Send Markdown or HTML, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in your bot's message.
- **caption\_entities** (`typing.Optional[typing.List[MessageEntity]]`) – List of special entities that appear in message text, which can be specified instead of `parse_mode`
- **reply\_markup** (`typing.Optional[types.InlineKeyboardMarkup]`) – A JSON-serialized object for an inline keyboard

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

**Return type** `typing.Union[types.Message, base.Boolean]`

```
async edit_media (media: aiogram.types.input_media.InputMedia, reply_markup: Optional[aiogram.types.inline_keyboard.InlineKeyboardMarkup] = None) → Union[aiogram.types.message.Message, Boolean]
```

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

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

Source <https://core.telegram.org/bots/api#editmessagemedia>

#### Parameters

- **media** (`types.InputMedia`) – A JSON-serialized object for a new media content of the message
- **reply\_markup** (`typing.Optional[types.InlineKeyboardMarkup]`) – A JSON-serialized object for a new inline keyboard

**Returns** On success, if the edited message was sent by the bot, the edited Message is returned, otherwise True is returned

**Return type** `typing.Union[types.Message, base.Boolean]`

**async edit\_reply\_markup** (*reply\_markup: Optional[aiogram.types.inline\_keyboard.InlineKeyboardMarkup] = None*) → `Union[aiogram.types.message.Message, Boolean]`

Use this method to edit only the reply markup of messages sent by the bot or via the bot (for inline bots).

Source: <https://core.telegram.org/bots/api#editmessagereplymarkup>

**Parameters** **reply\_markup** (`typing.Optional[types.InlineKeyboardMarkup]`) – A JSON-serialized object for an inline keyboard

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

**Return type** `typing.Union[types.Message, base.Boolean]`

**async delete\_reply\_markup** () → `Union[aiogram.types.message.Message, Boolean]`

Use this method to delete reply markup of messages sent by the bot or via the bot (for inline bots).

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

**Return type** `typing.Union[types.Message, base.Boolean]`

**async edit\_live\_location** (*latitude: Float, longitude: Float, reply\_markup: Optional[aiogram.types.inline\_keyboard.InlineKeyboardMarkup] = None*) → `Union[aiogram.types.message.Message, Boolean]`

Use this method to edit live location messages sent by the bot or via the bot (for inline bots). A location can be edited until its `live_period` expires or editing is explicitly disabled by a call to `stopMessageLiveLocation`.

Source: <https://core.telegram.org/bots/api#editmessagelivelocation>

#### Parameters

- **latitude** (`base.Float`) – Latitude of new location
- **longitude** (`base.Float`) – Longitude of new location
- **reply\_markup** (`typing.Optional[types.InlineKeyboardMarkup]`) – A JSON-serialized object for a new inline keyboard.

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

**Return type** `typing.Union[types.Message, base.Boolean]`

**async stop\_live\_location** (*reply\_markup: Optional[aiogram.types.inline\_keyboard.InlineKeyboardMarkup]* = None) → Union[*aiogram.types.message.Message*, Boolean]

Use this method to stop updating a live location message sent by the bot or via the bot (for inline bots) before live\_period expires.

Source: <https://core.telegram.org/bots/api#stopmessagelivelocation>

**Parameters** **reply\_markup** (typing.Optional[types.InlineKeyboardMarkup]) – A JSON-serialized object for a new inline keyboard.

**Returns** On success, if the message was sent by the bot, the sent Message is returned, otherwise True is returned.

**Return type** typing.Union[types.Message, base.Boolean]

**async delete** () → Boolean

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

Source: <https://core.telegram.org/bots/api#deletemessage>

**Returns** Returns True on success

**Return type** base.Boolean

**async pin** (*disable\_notification: Optional[Boolean] = None*) → Boolean

Use this method to add a message to the list of pinned messages in a chat. If the chat is not a private chat, the bot must be an administrator in the chat for this to work and must have the ‘can\_pin\_messages’ admin right in a supergroup or ‘can\_edit\_messages’ admin right in a channel. Returns True on success.

Source: <https://core.telegram.org/bots/api#pinchatmessage>

**Parameters** **disable\_notification** (typing.Optional[base.Boolean]) – Pass True, if it is not necessary to send a notification to all group members about the new pinned message

**Returns** Returns True on success

**Return type** base.Boolean

**async unpin** () → Boolean

Use this method to remove a message from the list of pinned messages in a chat. If the chat is not a private chat, the bot must be an administrator in the chat for this to work and must have the ‘can\_pin\_messages’ admin right in a supergroup or ‘can\_edit\_messages’ admin right in a channel. Returns True on success.

Source: <https://core.telegram.org/bots/api#unpinchatmessage>

**Returns** Returns True on success

**Return type** base.Boolean

**send\_copy** (*chat\_id: Union[str, int]*, *disable\_notification: Optional[bool] = None*, *disable\_web\_page\_preview: Optional[bool] = None*, *reply\_to\_message\_id: Optional[int] = None*, *allow\_sending\_without\_reply: Optional[Boolean] = None*, *reply\_markup: Optional[Union[aiogram.types.inline\_keyboard.InlineKeyboardMarkup, aiogram.types.reply\_keyboard.ReplyKeyboardMarkup]] = None*) → *aiogram.types.message.Message*

Send copy of current message

**Parameters**

- `chat_id` –
- `disable_notification` –
- `disable_web_page_preview` – for text messages only
- `reply_to_message_id` –
- `allow_sending_without_reply` –
- `reply_markup` –

**Returns**

## ContentType

**class** aiogram.types.message.ContentType

Bases: aiogram.utils.helper.Helper

List of message content types

WARNING: Single elements

**Key** TEXT

**Key** AUDIO

**Key** DOCUMENT

**Key** GAME

**Key** PHOTO

**Key** STICKER

**Key** VIDEO

**Key** VIDEO\_NOTE

**Key** VOICE

**Key** CONTACT

**Key** LOCATION

**Key** VENUE

**Key** NEW\_CHAT\_MEMBERS

**Key** LEFT\_CHAT\_MEMBER

**Key** INVOICE

**Key** SUCCESSFUL\_PAYMENT

**Key** CONNECTED\_WEBSITE

**Key** MIGRATE\_TO\_CHAT\_ID

**Key** MIGRATE\_FROM\_CHAT\_ID

**Key** UNKNOWN

**Key** ANY

## ContentTypes

**class** aiogram.types.message.**ContentTypes**

Bases: aiogram.utils.helper.Helper

List of message content types

WARNING: List elements.

**Key** TEXT

**Key** AUDIO

**Key** DOCUMENT

**Key** GAME

**Key** PHOTO

**Key** STICKER

**Key** VIDEO

**Key** VIDEO\_NOTE

**Key** VOICE

**Key** CONTACT

**Key** LOCATION

**Key** VENUE

**Key** NEW\_CHAT\_MEMBERS

**Key** LEFT\_CHAT\_MEMBER

**Key** INVOICE

**Key** SUCCESSFUL\_PAYMENT

**Key** CONNECTED\_WEBSITE

**Key** MIGRATE\_TO\_CHAT\_ID

**Key** MIGRATE\_FROM\_CHAT\_ID

**Key** UNKNOWN

**Key** ANY

## ParseMode

**class** aiogram.types.message.**ParseMode**

Bases: aiogram.utils.helper.Helper

Parse modes

**Key** MARKDOWN

**Key** HTML



## MaskPosition

**class** aiogram.types.mask\_position.**MaskPosition** (*conf: Optional[Dict[str, Any]] = None, \*\*kwargs: Any*)

Bases: *aiogram.types.base.TelegramObject*

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

<https://core.telegram.org/bots/api#maskposition>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

## UserProfilePhotos

**class** aiogram.types.user\_profile\_photos.**UserProfilePhotos** (*conf: Optional[Dict[str, Any]] = None, \*\*kwargs: Any*)

Bases: *aiogram.types.base.TelegramObject*

This object represent a user's profile pictures.

<https://core.telegram.org/bots/api#userprofilephotos>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

## Invoice

**class** aiogram.types.invoice.**Invoice** (*conf: Optional[Dict[str, Any]] = None, \*\*kwargs: Any*)

Bases: *aiogram.types.base.TelegramObject*

This object contains basic information about an invoice.

<https://core.telegram.org/bots/api#invoice>

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

## AuthWidgetData

```
class aiogram.types.auth_widget_data.AuthWidgetData (conf: Optional[Dict[str, Any]] =  
None, **kwargs: Any)
```

Bases: *aiogram.types.base.TelegramObject*

Deserialize object

### Parameters

- **conf** –
- **kwargs** –

```
classmethod parse (request: aiohttp.web_request.Request) →  
aiogram.types.auth_widget_data.AuthWidgetData
```

Parse request as Telegram auth widget data.

**Parameters** **request** –

**Returns** *AuthWidgetData*

**Raise** *aiohttp.web.HTTPBadRequest*

## 4.5 Dispatcher

### 4.5.1 Filters

#### Basics

Filter factory greatly simplifies the reuse of filters when registering handlers.

#### Filters factory

```
class aiogram.dispatcher.filters.FiltersFactory (dispatcher)
```

Bases: *object*

Filters factory

```
bind (callback: Union[Callable, aiogram.dispatcher.filters.filters.AbstractFilter], validator: Op-  
tional[Callable] = None, event_handlers: Optional[List[aiogram.dispatcher.handler.Handler]]  
= None, exclude_event_handlers: Optional[Iterable[aiogram.dispatcher.handler.Handler]] =  
None)
```

Register filter

### Parameters

- **callback** – callable or subclass of *AbstractFilter*
- **validator** – custom validator.
- **event\_handlers** – list of instances of *Handler*
- **exclude\_event\_handlers** – list of excluded event handlers (*Handler*)

```
unbind (callback: Union[Callable, aiogram.dispatcher.filters.filters.AbstractFilter])
```

Unregister filter

**Parameters** **callback** – callable of subclass of *AbstractFilter*

**resolve** (*event\_handler*, *\*custom\_filters*, *\*\*full\_config*) → List[Union[Callable, aiogram.dispatcher.filters.filters.AbstractFilter]]  
 Resolve filters to filters-set

#### Parameters

- **event\_handler** –
- **custom\_filters** –
- **full\_config** –

#### Returns

### Builtin filters

aiogram has some builtin filters. Here you can see all of them:

### Command

```
class aiogram.dispatcher.filters.Command(commands: Union[Iterable, str], prefixes: Union[Iterable, str] = '/', ignore_case: bool = True, ignore_mention: bool = False, ignore_caption: bool = True)

Bases: aiogram.dispatcher.filters.filters.Filter
```

You can handle commands by using this filter.

If filter is successful processed the *Command.CommandObj* will be passed to the handler arguments.

By default this filter is registered for messages and edited messages handlers.

Filter can be initialized from filters factory or by simply creating instance of this class.

Examples:

```
@dp.message_handler(commands=['myCommand'])
@dp.message_handler(Command(['myCommand']))
@dp.message_handler(commands=['myCommand'], commands_prefix='!/')
```

#### Parameters

- **commands** – Command or list of commands always without leading slashes (prefix)
- **prefixes** – Allowed commands prefix. By default is slash. If you change the default behavior pass the list of prefixes to this argument.
- **ignore\_case** – Ignore case of the command
- **ignore\_mention** – Ignore mention in command (By default this filter pass only the commands addressed to current bot)
- **ignore\_caption** – Ignore caption from message (in message types like photo, video, audio, etc) By default is True. If you want check commands in captions, you also should set required content\_types.

Examples:

```
@dp.message_handler(commands=['myCommand'], commands_ignore_
↳caption=False, content_types=ContentType.ANY)
@dp.message_handler(Command(['myCommand'], ignore_caption=False),
↳content_types=[ContentType.TEXT, ContentType.DOCUMENT])
```

**classmethod validate** (*full\_config: Dict[str, Any]*) → Optional[Dict[str, Any]]

Validator for filters factory

From filters factory this filter can be registered with arguments:

- `command`
- `commands_prefix` (will be passed as `prefixes`)
- `commands_ignore_mention` (will be passed as `ignore_mention`)
- `commands_ignore_caption` (will be passed as `ignore_caption`)

**Parameters full\_config** –

**Returns** config or empty dict

**async check** (*message: aiogram.types.message.Message*)

Will be called when filters checks.

This method must be overridden.

**Parameters args** –

**Returns**

**class CommandObj** (*prefix: str = '/', command: str = "", mention: str = None, args: str = None*)

Bases: object

Instance of this object is always has command and it prefix.

Can be passed as keyword argument `command` to the handler

Instance of this object is always has command and it prefix.

Can be passed as keyword argument `command` to the handler

**prefix: str = '/'**

Command without prefix and mention

**command: str = ''**

Mention (if available)

**mention: str = None**

Command argument

**args: str = None**

**property mentioned**

This command has mention?

**Returns**

**property text**

Generate original text from object

**Returns**

## CommandStart

```
class aiogram.dispatcher.filters.CommandStart (deep_link: Optional[Union[str, Pattern[str]]] = None, encoded: bool = False)
```

Bases: aiogram.dispatcher.filters.builtin.Command

This filter based on [Command](#) filter but can handle only /start command.

Also this filter can handle [deep-linking](#) arguments.

Example:

```
@dp.message_handler(CommandStart(re.compile(r'ref-([\d]+)')))
```

### Parameters

- **deep\_link** – string or compiled regular expression (by `re.compile(...)`).
- **encoded** – set True if you're waiting for encoded payload (default - False).

```
async check (message: aiogram.types.message.Message)
```

If deep-linking is passed to the filter result of the matching will be passed as `deep_link` to the handler

**Parameters** `message` –

**Returns**

## CommandHelp

```
class aiogram.dispatcher.filters.CommandHelp
```

Bases: aiogram.dispatcher.filters.builtin.Command

This filter based on [Command](#) filter but can handle only /help command.

Filter can be initialized from filters factory or by simply creating instance of this class.

Examples:

```
@dp.message_handler(commands=['myCommand'])
@dp.message_handler(Command(['myCommand']))
@dp.message_handler(commands=['myCommand'], commands_prefix='!/')
```

### Parameters

- **commands** – Command or list of commands always without leading slashes (prefix)
- **prefixes** – Allowed commands prefix. By default is slash. If you change the default behavior pass the list of prefixes to this argument.
- **ignore\_case** – Ignore case of the command
- **ignore\_mention** – Ignore mention in command (By default this filter pass only the commands addressed to current bot)
- **ignore\_caption** – Ignore caption from message (in message types like photo, video, audio, etc) By default is True. If you want check commands in captions, you also should set `required_content_types`.

Examples:

```
@dp.message_handler(commands=['myCommand'], commands_ignore_
↳caption=False, content_types=ContentType.ANY)
@dp.message_handler(Command(['myCommand'], ignore_caption=False),
↳content_types=[ContentType.TEXT, ContentType.DOCUMENT])
```

## CommandSettings

**class** aiogram.dispatcher.filters.**CommandSettings**

Bases: aiogram.dispatcher.filters.builtin.Command

This filter based on [Command](#) filter but can handle only /settings command.

Filter can be initialized from filters factory or by simply creating instance of this class.

Examples:

```
@dp.message_handler(commands=['myCommand'])
@dp.message_handler(Command(['myCommand']))
@dp.message_handler(commands=['myCommand'], commands_prefix='!/')
```

### Parameters

- **commands** – Command or list of commands always without leading slashes (prefix)
- **prefixes** – Allowed commands prefix. By default is slash. If you change the default behavior pass the list of prefixes to this argument.
- **ignore\_case** – Ignore case of the command
- **ignore\_mention** – Ignore mention in command (By default this filter pass only the commands addressed to current bot)
- **ignore\_caption** – Ignore caption from message (in message types like photo, video, audio, etc) By default is True. If you want check commands in captions, you also should set required content\_types.

Examples:

```
@dp.message_handler(commands=['myCommand'], commands_ignore_
↳caption=False, content_types=ContentType.ANY)
@dp.message_handler(Command(['myCommand'], ignore_caption=False),
↳content_types=[ContentType.TEXT, ContentType.DOCUMENT])
```

## CommandPrivacy

**class** aiogram.dispatcher.filters.**CommandPrivacy**

Bases: aiogram.dispatcher.filters.builtin.Command

This filter based on [Command](#) filter but can handle only /privacy command.

Filter can be initialized from filters factory or by simply creating instance of this class.

Examples:

```
@dp.message_handler(commands=['myCommand'])
@dp.message_handler(Command(['myCommand']))
@dp.message_handler(commands=['myCommand'], commands_prefix='!/')
```

### Parameters

- **commands** – Command or list of commands always without leading slashes (prefix)
- **prefixes** – Allowed commands prefix. By default is slash. If you change the default behavior pass the list of prefixes to this argument.
- **ignore\_case** – Ignore case of the command
- **ignore\_mention** – Ignore mention in command (By default this filter pass only the commands addressed to current bot)
- **ignore\_caption** – Ignore caption from message (in message types like photo, video, audio, etc) By default is True. If you want check commands in captions, you also should set required content\_types.

Examples:

```
@dp.message_handler(commands=['myCommand'], commands_ignore_
↳caption=False, content_types=ContentType.ANY)
@dp.message_handler(Command(['myCommand'], ignore_caption=False),
↳content_types=[ContentType.TEXT, ContentType.DOCUMENT])
```

### Text

```
class aiogram.dispatcher.filters.Text (equals: Optional[Union[str, babel.support.LazyProxy,
Iterable[Union[str, babel.support.LazyProxy]]]]
= None, contains: Optional[Union[str, ba-
bel.support.LazyProxy, Iterable[Union[str, ba-
bel.support.LazyProxy]]]] = None, startswith:
Optional[Union[str, babel.support.LazyProxy, It-
erable[Union[str, babel.support.LazyProxy]]]]
= None, endswith: Optional[Union[str, ba-
bel.support.LazyProxy, Iterable[Union[str,
babel.support.LazyProxy]]]] = None, ig-
nore_case=False)
```

Bases: aiogram.dispatcher.filters.filters.Filter

Simple text filter

Check text for one of pattern. Only one mode can be used in one filter. In every pattern, a single string is treated as a list with 1 element.

### Parameters

- **equals** – True if object's text in the list
- **contains** – True if object's text contains all strings from the list
- **startswith** – True if object's text starts with any of strings from the list
- **endswith** – True if object's text ends with any of strings from the list
- **ignore\_case** – case insensitive

**classmethod validate** (full\_config: Dict[str, Any])

Here method `validate` is optional. If you need to use filter from filters factory you need to override this method.

**Parameters** **full\_config** – dict with arguments passed to handler registrar

**Returns** Current filter config

**async check** (*obj: Union[aiogram.types.message.Message, aiogram.types.callback\_query.CallbackQuery, aiogram.types.inline\_query.InlineQuery, aiogram.types.poll.Poll]*)

Will be called when filters checks.

This method must be overridden.

**Parameters** **args** –

**Returns**

## HashTag

**class** aiogram.dispatcher.filters.**HashTag** (*hashtags=None, cashtags=None*)

Bases: aiogram.dispatcher.filters.filters.Filter

Filter for hashtag's and cashtag's

**classmethod validate** (*full\_config: Dict[str, Any]*)

Here method `validate` is optional. If you need to use filter from filters factory you need to override this method.

**Parameters** **full\_config** – dict with arguments passed to handler registrar

**Returns** Current filter config

**async check** (*message: aiogram.types.message.Message*)

Will be called when filters checks.

This method must be overridden.

**Parameters** **args** –

**Returns**

## Regexp

**class** aiogram.dispatcher.filters.**Regexp** (*regexp*)

Bases: aiogram.dispatcher.filters.filters.Filter

Regexp filter for messages and callback query

**classmethod validate** (*full\_config: Dict[str, Any]*)

Here method `validate` is optional. If you need to use filter from filters factory you need to override this method.

**Parameters** **full\_config** – dict with arguments passed to handler registrar

**Returns** Current filter config

**async check** (*obj: Union[aiogram.types.message.Message, aiogram.types.callback\_query.CallbackQuery, aiogram.types.inline\_query.InlineQuery, aiogram.types.poll.Poll]*)

Will be called when filters checks.

This method must be overridden.

**Parameters** **args** –

**Returns**



### RegexpCommandsFilter

**class** aiogram.dispatcher.filters.**RegexpCommandsFilter** (*regex\_commands*)

Bases: aiogram.dispatcher.filters.filters.BoundFilter

Check commands by regexp in message

**async check** (*message*)

Will be called when filters checks.

This method must be overridden.

**Parameters** *args* –

**Returns**

### ContentTypeFilter

**class** aiogram.dispatcher.filters.**ContentTypeFilter** (*content\_types*)

Bases: aiogram.dispatcher.filters.filters.BoundFilter

Check message content type

**async check** (*message*)

Will be called when filters checks.

This method must be overridden.

**Parameters** *args* –

**Returns**

### IsSenderContact

**class** aiogram.dispatcher.filters.**IsSenderContact** (*is\_sender\_contact: bool*)

Bases: aiogram.dispatcher.filters.filters.BoundFilter

Filter check that the contact matches the sender

*is\_sender\_contact=True* - contact matches the sender *is\_sender\_contact=False* - result will be inverted

**async check** (*message: aiogram.types.message.Message*) → bool

Will be called when filters checks.

This method must be overridden.

**Parameters** *args* –

**Returns**

## StateFilter

**class** aiogram.dispatcher.filters.**StateFilter** (*dispatcher, state*)

Bases: aiogram.dispatcher.filters.filters.BoundFilter

Check user state

**async check** (*obj*)

Will be called when filters checks.

This method must be overridden.

**Parameters** **args** –

**Returns**

## ExceptionsFilter

**class** aiogram.dispatcher.filters.**ExceptionsFilter** (*exception*)

Bases: aiogram.dispatcher.filters.filters.BoundFilter

Filter for exceptions

**async check** (*update, exception*)

Will be called when filters checks.

This method must be overridden.

**Parameters** **args** –

**Returns**

## IDFilter

**class** aiogram.dispatcher.filters.builtin.**IDFilter** (*user\_id: Optional[Union[Iterable[Union[int, str]], str, int]] = None, chat\_id: Optional[Union[Iterable[Union[int, str]], str, int]] = None*)

Bases: aiogram.dispatcher.filters.filters.Filter

**Parameters**

• **user\_id** –

• **chat\_id** –

**classmethod validate** (*full\_config: Dict[str, Any]*) → Optional[Dict[str, Any]]

Here method `validate` is optional. If you need to use filter from filters factory you need to override this method.

**Parameters** **full\_config** – dict with arguments passed to handler registrar

**Returns** Current filter config

**async check** (*obj: Union[aiogram.types.message.Message, aiogram.types.callback\_query.CallbackQuery, aiogram.types.inline\_query.InlineQuery]*)

Will be called when filters checks.

This method must be overridden.

**Parameters** `args` –

**Returns**

### AdminFilter

```
class aiogram.dispatcher.filters.AdminFilter (is_chat_admin: Optional[Union[Iterable[Union[int, str, int, bool]] = None], Op-  
str]],
```

Bases: aiogram.dispatcher.filters.filters.Filter

Checks if user is admin in a chat. If `is_chat_admin` is not set, the filter will check in the current chat (correct only for messages). `is_chat_admin` is required for `InlineQuery`.

**classmethod** `validate` (*full\_config: Dict[str, Any]*) → Optional[Dict[str, Any]]

Here method `validate` is optional. If you need to use filter from filters factory you need to override this method.

**Parameters** `full_config` – dict with arguments passed to handler registrar

**Returns** Current filter config

```
async check (obj: Union[aiogram.types.message.Message, aiogram.types.callback_query.CallbackQuery,  
aiogram.types.inline_query.InlineQuery]) → bool
```

Will be called when filters checks.

This method must be overridden.

**Parameters** `args` –

**Returns**

### IsReplyFilter

```
class aiogram.dispatcher.filters.IsReplyFilter (is_reply)  
Bases: aiogram.dispatcher.filters.filters.BoundFilter
```

Check if message is replied and send reply message to handler

```
async check (msg: aiogram.types.message.Message)
```

Will be called when filters checks.

This method must be overridden.

**Parameters** `args` –

**Returns**

### ForwardedMessageFilter

```
class aiogram.dispatcher.filters.ForwardedMessageFilter (is_forwarded: bool)  
Bases: aiogram.dispatcher.filters.filters.BoundFilter
```

```
async check (message: aiogram.types.message.Message)
```

Will be called when filters checks.

This method must be overridden.

**Parameters** `args` –

## Returns

### ChatTypeFilter

```
class aiogram.dispatcher.filters.ChatTypeFilter(chat_type: Container[aiogram.types.chat.ChatType])
    Bases: aiogram.dispatcher.filters.filters.BoundFilter

    async check (obj: Union[aiogram.types.message.Message, aiogram.types.callback_query.CallbackQuery])
        Will be called when filters checks.

        This method must be overridden.

    Parameters args –

    Returns
```

### Making own filters (Custom filters)

Own filter can be:

- any callable object
- any async function
- any anonymous function (Example: `lambda msg: msg.text == 'spam'`)
- Subclass of `AbstractFilter`, `Filter` or `BoundFilter`

### AbstractFilter

```
class aiogram.dispatcher.filters.AbstractFilter
    Bases: abc.ABC

    Abstract class for custom filters.

    abstract classmethod validate (full_config: Dict[str, Any]) → Optional[Dict[str, Any]]
        Validate and parse config.

        This method will be called by the filters factory when you bind this filter. Must be overridden.

        Parameters full_config – dict with arguments passed to handler registrar

        Returns Current filter config

    abstract async check (*args) → bool
        Will be called when filters checks.

        This method must be overridden.

        Parameters args –

        Returns
```

## Filter

**class** aiogram.dispatcher.filters.**Filter**

Bases: aiogram.dispatcher.filters.filters.AbstractFilter

You can make subclasses of that class for custom filters.

Method `check` must be overridden

**classmethod** **validate** (*full\_config: Dict[str, Any]*) → Optional[Dict[str, Any]]

Here method `validate` is optional. If you need to use filter from filters factory you need to override this method.

**Parameters** **full\_config** – dict with arguments passed to handler registrar

**Returns** Current filter config

## BoundFilter

**class** aiogram.dispatcher.filters.**BoundFilter**

Bases: aiogram.dispatcher.filters.filters.Filter

To easily create your own filters with one parameter, you can inherit from this filter.

You need to implement `__init__` method with single argument related with key attribute and `check` method where you need to implement filter logic.

**key = None**

Unique name of the filter argument. You need to override this attribute.

**required = False**

If `True` this filter will be added to the all of the registered handlers

**default = None**

Default value for configure required filters

**classmethod** **validate** (*full\_config: Dict[str, Any]*) → Dict[str, Any]

If `cls.key` is not `None` and that is in config returns config with that argument.

**Parameters** **full\_config** –

**Returns**

```
class ChatIdFilter(BoundFilter):
    key = 'chat_id'

    def __init__(self, chat_id: typing.Union[typing.Iterable, int]):
        if isinstance(chat_id, int):
            chat_id = [chat_id]
        self.chat_id = chat_id

    def check(self, message: types.Message) -> bool:
        return message.chat.id in self.chat_id

dp.filters_factory.bind(ChatIdFilter, event_handlers=[dp.message_handlers])
```

## 4.5.2 Finite state machine

### Storage

Coming soon...

### Available storage's

Coming soon...

### Memory storage

```
class aiogram.contrib.fsm_storage.memory.MemoryStorage
```

Bases: aiogram.dispatcher.storage.BaseStorage

In-memory based states storage.

This type of storage is not recommended for usage in bots, because you will lost all states after restarting.

### Redis storage

```
class aiogram.contrib.fsm_storage.redis.RedisStorage (host='localhost', port=6379,  
                                                    db=None, password=None,  
                                                    ssl=None, loop=None,  
                                                    **kwargs)
```

Bases: aiogram.dispatcher.storage.BaseStorage

Simple Redis-base storage for FSM.

Usage:

```
storage = RedisStorage('localhost', 6379, db=5)  
dp = Dispatcher(bot, storage=storage)
```

And need to close Redis connection when shutdown

```
await dp.storage.close()  
await dp.storage.wait_closed()
```

### Mongo storage

```
class aiogram.contrib.fsm_storage.mongo.MongoStorage (host='localhost', port=27017,  
                                                       db_name='aiogram_fsm',  
                                                       uri=None, username=None,  
                                                       password=None, index=True,  
                                                       **kwargs)
```

Bases: aiogram.dispatcher.storage.BaseStorage

Mongo-based storage for FSM.

Usage:

```
storage = MongoStorage(host='localhost', port=27017, db_name='aiogram_fsm')
dp = Dispatcher(bot, storage=storage)
```

And need to close Mongo client connections when shutdown

```
await dp.storage.close()
await dp.storage.wait_closed()
```

## Rethink DB storage

```
class aiogram.contrib.fsm_storage.rethinkdb.RethinkDBStorage (host: str = 'local-
    host', port: int =
    28015, db: str =
    'aiogram', table:
    str = 'aiogram',
    auth_key: Op-
    tional[str] = None,
    user: Optional[str]
    = None, password:
    Optional[str] =
    None, timeout:
    int = 20, ssl:
    Optional[dict] =
    None, loop: Op-
    tional[asyncio.events.AbstractEventLoop]
    = None)
```

Bases: aiogram.dispatcher.storage.BaseStorage

RethinkDB-based storage for FSM.

Usage:

```
storage = RethinkDBStorage(db='aiogram', table='aiogram', user='aiogram',
    ↪password='aiogram_secret')
dispatcher = Dispatcher(bot, storage=storage)
```

And need to close connection when shutdown

```
await storage.close()
await storage.wait_closed()
```

## Making own storage's

Coming soon...

## States

Coming soon...

## State utils

Coming soon...

## State

Coming soon...

## States group

Coming soon...

## 4.5.3 Middleware

### Bases

Coming soon...

### Making own middleware's

Coming soon...

### Available middleware's

Coming soon...

## 4.5.4 Webhook

Coming soon...

### Bases

Coming soon...



## Security

Coming soon...

## Making requests when getting updates

Coming soon...

## 4.5.5 Basics

Coming soon...

## 4.5.6 Available handlers

Coming soon...

## Handler class

Coming soon...

## 4.5.7 Features

Coming soon...

## 4.5.8 Dispatcher class

```
class aiogram.Dispatcher(bot, loop=None, storage: Optional[aiogram.dispatcher.storage.BaseStorage]
                        = None, run_tasks_by_default: bool = False, throttling_rate_limit=0.1,
                        no_throttle_error=False, filters_factory=None)
    Bases: aiogram.utils.mixins.DataMixin, aiogram.utils.mixins.ContextInstanceMixin
```

Simple Updates dispatcher

It will process incoming updates: messages, edited messages, channel posts, edited channel posts, inline queries, chosen inline results, callback queries, shipping queries, pre-checkout queries.

**async skip\_updates()**

You can skip old incoming updates from queue. This method is not recommended to use if you use payments or you bot has high-load.

**Returns** None

**async process\_updates** (*updates, fast: Optional[bool] = True*)

Process list of updates

**Parameters**

- **updates** –
- **fast** –

**Returns**

**async process\_update** (*update: aiogram.types.update.Update*)

Process single update object

**Parameters** *update* –

**Returns**

**async reset\_webhook** (*check=True*) → bool

Reset webhook

**Parameters** *check* – check before deleting

**Returns**

**async start\_polling** (*timeout=20, relax=0.1, limit=None, reset\_webhook=None, fast: Optional[bool] = True, error\_sleep: int = 5*)

Start long-polling

**Parameters**

- *timeout* –
- *relax* –
- *limit* –
- *reset\_webhook* –
- *fast* –

**Returns**

**stop\_polling** ()

Break long-polling process.

**Returns**

**async wait\_closed** ()

Wait for the long-polling to close

**Returns**

**is\_polling** ()

Check if polling is enabled

**Returns**

**register\_message\_handler** (*callback, \*custom\_filters, commands=None, regexp=None, content\_types=None, state=None, run\_task=None, \*\*kwargs*)

Register handler for message

```
# This handler works only if state is None (by default).
dp.register_message_handler(cmd_start, commands=['start', 'about'])
dp.register_message_handler(entry_point, commands=['setup'])

# This handler works only if current state is "first_step"
dp.register_message_handler(step_handler_1, state="first_step")

# If you want to handle all states by one handler, use `state="*"`.
dp.register_message_handler(cancel_handler, commands=['cancel'], state="*")
dp.register_message_handler(cancel_handler, lambda msg: msg.text.lower() ==
↪ 'cancel', state="*")
```

**Parameters**

- **callback** –
- **commands** – list of commands
- **regexp** – REGEXP
- **content\_types** – List of content types.
- **custom\_filters** – list of custom filters
- **kwargs** –
- **state** –

**Returns** decorated function

**message\_handler** (\*custom\_filters, commands=None, regexp=None, content\_types=None, state=None, run\_task=None, \*\*kwargs)  
 Decorator for message handler

Examples:

Simple commands handler:

```
@dp.message_handler(commands=['start', 'welcome', 'about'])
async def cmd_handler(message: types.Message):
```

Filter messages by regular expression:

```
@dp.message_handler(regexp='^[a-z]+-[0-9]+')
async def msg_handler(message: types.Message):
```

Filter messages by command regular expression:

```
@dp.message_handler(filters.RegexpCommandsFilter(regexp_commands=['item_([0-9]*)']))
async def send_welcome(message: types.Message):
```

Filter by content type:

```
@dp.message_handler(content_types=ContentType.PHOTO | ContentType.DOCUMENT)
async def audio_handler(message: types.Message):
```

Filter by custom function:

```
@dp.message_handler(lambda message: message.text and 'hello' in message.text.lower())
async def text_handler(message: types.Message):
```

Use multiple filters:

```
@dp.message_handler(commands=['command'], content_types=ContentType.TEXT)
async def text_handler(message: types.Message):
```

Register multiple filters set for one handler:

```
@dp.message_handler(commands=['command'])
@dp.message_handler(lambda message: demojize(message.text) == ':new_moon_with_face:')
async def text_handler(message: types.Message):
```

This handler will be called if the message starts with `/command` OR is some emoji

By default `content_type` is `ContentType.TEXT`

#### Parameters

- **commands** – list of commands
- **regexp** – REGEXP
- **content\_types** – List of content types.
- **custom\_filters** – list of custom filters
- **kwargs** –
- **state** –
- **run\_task** – run callback in task (no wait results)

**Returns** decorated function

```
register_edited_message_handler (callback, *custom_filters, commands=None, reg-  
exp=None, content_types=None, state=None,  
run_task=None, **kwargs)
```

Register handler for edited message

#### Parameters

- **callback** –
- **commands** – list of commands
- **regexp** – REGEXP
- **content\_types** – List of content types.
- **state** –
- **custom\_filters** – list of custom filters
- **run\_task** – run callback in task (no wait results)
- **kwargs** –

**Returns** decorated function

```
edited_message_handler (*custom_filters, commands=None, regexp=None, content_types=None,  
state=None, run_task=None, **kwargs)
```

Decorator for edited message handler

You can use combination of different handlers

```
@dp.message_handler()  
@dp.edited_message_handler()  
async def msg_handler(message: types.Message):
```

#### Parameters

- **commands** – list of commands
- **regexp** – REGEXP
- **content\_types** – List of content types.
- **state** –
- **custom\_filters** – list of custom filters

- **run\_task** – run callback in task (no wait results)
- **kwargs** –

**Returns** decorated function

**register\_channel\_post\_handler** (*callback, \*custom\_filters, commands=None, regexp=None, content\_types=None, state=None, run\_task=None, \*\*kwargs*)

Register handler for channel post

#### Parameters

- **callback** –
- **commands** – list of commands
- **regexp** – REGEXP
- **content\_types** – List of content types.
- **state** –
- **custom\_filters** – list of custom filters
- **run\_task** – run callback in task (no wait results)
- **kwargs** –

**Returns** decorated function

**channel\_post\_handler** (*\*custom\_filters, commands=None, regexp=None, content\_types=None, state=None, run\_task=None, \*\*kwargs*)

Decorator for channel post handler

#### Parameters

- **commands** – list of commands
- **regexp** – REGEXP
- **content\_types** – List of content types.
- **state** –
- **custom\_filters** – list of custom filters
- **run\_task** – run callback in task (no wait results)
- **kwargs** –

**Returns** decorated function

**register\_edited\_channel\_post\_handler** (*callback, \*custom\_filters, commands=None, regexp=None, content\_types=None, state=None, run\_task=None, \*\*kwargs*)

Register handler for edited channel post

#### Parameters

- **callback** –
- **commands** – list of commands
- **regexp** – REGEXP
- **content\_types** – List of content types.
- **state** –

- **custom\_filters** – list of custom filters
- **run\_task** – run callback in task (no wait results)
- **kwargs** –

**Returns** decorated function

**edited\_channel\_post\_handler** (*\*custom\_filters, commands=None, regexp=None, content\_types=None, state=None, run\_task=None, \*\*kwargs*)

Decorator for edited channel post handler

**Parameters**

- **commands** – list of commands
- **regexp** – REGEXP
- **content\_types** – List of content types.
- **custom\_filters** – list of custom filters
- **state** –
- **run\_task** – run callback in task (no wait results)
- **kwargs** –

**Returns** decorated function

**register\_inline\_handler** (*callback, \*custom\_filters, state=None, run\_task=None, \*\*kwargs*)

Register handler for inline query

Example:

```
dp.register_inline_handler(some_inline_handler, lambda inline_query: True)
```

**Parameters**

- **callback** –
- **custom\_filters** – list of custom filters
- **state** –
- **run\_task** – run callback in task (no wait results)
- **kwargs** –

**Returns** decorated function

**inline\_handler** (*\*custom\_filters, state=None, run\_task=None, \*\*kwargs*)

Decorator for inline query handler

Example:

```
@dp.inline_handler(lambda inline_query: True)
async def some_inline_handler(inline_query: types.InlineQuery)
```

**Parameters**

- **state** –
- **custom\_filters** – list of custom filters
- **run\_task** – run callback in task (no wait results)

- **kwargs** –

**Returns** decorated function

**register\_chosen\_inline\_handler** (*callback*, *\*custom\_filters*, *state=None*, *run\_task=None*, *\*\*kwargs*)

Register handler for chosen inline query

Example:

```
dp.register_chosen_inline_handler(some_chosen_inline_handler, lambda chosen_
↳ inline_query: True)
```

#### Parameters

- **callback** –
- **state** –
- **custom\_filters** –
- **run\_task** – run callback in task (no wait results)
- **kwargs** –

#### Returns

**chosen\_inline\_handler** (*\*custom\_filters*, *state=None*, *run\_task=None*, *\*\*kwargs*)

Decorator for chosen inline query handler

Example:

```
@dp.chosen_inline_handler(lambda chosen_inline_query: True)
async def some_chosen_inline_handler(chosen_inline_query: types.
↳ ChosenInlineResult)
```

#### Parameters

- **state** –
- **custom\_filters** –
- **run\_task** – run callback in task (no wait results)
- **kwargs** –

#### Returns

**register\_callback\_query\_handler** (*callback*, *\*custom\_filters*, *state=None*, *run\_task=None*, *\*\*kwargs*)

Register handler for callback query

Example:

```
dp.register_callback_query_handler(some_callback_handler, lambda callback_
↳ query: True)
```

#### Parameters

- **callback** –
- **state** –

- **custom\_filters** –
- **run\_task** – run callback in task (no wait results)
- **kwargs** –

**callback\_query\_handler** (*\*custom\_filters, state=None, run\_task=None, \*\*kwargs*)

Decorator for callback query handler

Example:

```
@dp.callback_query_handler(lambda callback_query: True)  
async def some_callback_handler(callback_query: types.CallbackQuery)
```

#### Parameters

- **state** –
- **custom\_filters** –
- **run\_task** – run callback in task (no wait results)
- **kwargs** –

**register\_shipping\_query\_handler** (*callback, \*custom\_filters, state=None, run\_task=None, \*\*kwargs*)

Register handler for shipping query

Example:

```
dp.register_shipping_query_handler(some_shipping_query_handler, lambda_  
↪shipping_query: True)
```

#### Parameters

- **callback** –
- **state** –
- **custom\_filters** –
- **run\_task** – run callback in task (no wait results)
- **kwargs** –

**shipping\_query\_handler** (*\*custom\_filters, state=None, run\_task=None, \*\*kwargs*)

Decorator for shipping query handler

Example:

```
@dp.shipping_query_handler(lambda shipping_query: True)  
async def some_shipping_query_handler(shipping_query: types.ShippingQuery)
```

#### Parameters

- **state** –
- **custom\_filters** –
- **run\_task** – run callback in task (no wait results)
- **kwargs** –



**register\_pre\_checkout\_query\_handler** (*callback*, *\*custom\_filters*, *state=None*, *run\_task=None*, *\*\*kwargs*)

Register handler for pre-checkout query

Example:

```
dp.register_pre_checkout_query_handler(some_pre_checkout_query_handler,
↳ lambda shipping_query: True)
```

#### Parameters

- **callback** –
- **state** –
- **custom\_filters** –
- **run\_task** – run callback in task (no wait results)
- **kwargs** –

**pre\_checkout\_query\_handler** (*\*custom\_filters*, *state=None*, *run\_task=None*, *\*\*kwargs*)

Decorator for pre-checkout query handler

Example:

```
@dp.pre_checkout_query_handler(lambda shipping_query: True)
async def some_pre_checkout_query_handler(shipping_query: types.ShippingQuery)
```

#### Parameters

- **state** –
- **custom\_filters** –
- **run\_task** – run callback in task (no wait results)
- **kwargs** –

**register\_poll\_handler** (*callback*, *\*custom\_filters*, *run\_task=None*, *\*\*kwargs*)

Register handler for poll

Example:

```
dp.register_poll_handler(some_poll_handler)
```

#### Parameters

- **callback** –
- **custom\_filters** –
- **run\_task** – run callback in task (no wait results)
- **kwargs** –

**poll\_handler** (*\*custom\_filters*, *run\_task=None*, *\*\*kwargs*)

Decorator for poll handler

Example:

```
@dp.poll_handler()
async def some_poll_handler(poll: types.Poll)
```

#### Parameters

- **custom\_filters** –
- **run\_task** – run callback in task (no wait results)
- **kwargs** –

**register\_poll\_answer\_handler** (*callback*, *\*custom\_filters*, *run\_task=None*, *\*\*kwargs*)

Register handler for poll\_answer

Example:

```
dp.register_poll_answer_handler(some_poll_answer_handler)
```

#### Parameters

- **callback** –
- **custom\_filters** –
- **run\_task** – run callback in task (no wait results)
- **kwargs** –

**poll\_answer\_handler** (*\*custom\_filters*, *run\_task=None*, *\*\*kwargs*)

Decorator for poll\_answer handler

Example:

```
@dp.poll_answer_handler()
async def some_poll_answer_handler(poll_answer: types.PollAnswer)
```

#### Parameters

- **custom\_filters** –
- **run\_task** – run callback in task (no wait results)
- **kwargs** –

**register\_errors\_handler** (*callback*, *\*custom\_filters*, *exception=None*, *run\_task=None*, *\*\*kwargs*)

Register handler for errors

#### Parameters

- **callback** –
- **exception** – you can make handler for specific errors type
- **run\_task** – run callback in task (no wait results)

**errors\_handler** (*\*custom\_filters*, *exception=None*, *run\_task=None*, *\*\*kwargs*)

Decorator for errors handler

#### Parameters

- **exception** – you can make handler for specific errors type

- **run\_task** – run callback in task (no wait results)

#### Returns

**current\_state** (\*, chat: *Optional[Union[str, int]] = None*, user: *Optional[Union[str, int]] = None*)

→ aiogram.dispatcher.storage.FSMContext

Get current state for user in chat as context

```
with dp.current_state(chat=message.chat.id, user=message.user.id) as state:
    pass

state = dp.current_state()
state.set_state('my_state')
```

#### Parameters

- **chat** –
- **user** –

#### Returns

**async throttle** (key, \*, rate=None, user\_id=None, chat\_id=None, no\_error=None) → bool

Execute throttling manager. Returns True if limit has not exceeded otherwise raises ThrottleError or returns False

#### Parameters

- **key** – key in storage
- **rate** – limit (by default is equal to default rate limit)
- **user\_id** – user id
- **chat\_id** – chat id
- **no\_error** – return boolean value instead of raising error

#### Returns

bool

**async check\_key** (key, chat\_id=None, user\_id=None)

Get information about key in bucket

#### Parameters

- **key** –
- **chat\_id** –
- **user\_id** –

#### Returns

**async release\_key** (key, chat\_id=None, user\_id=None)

Release blocked key

#### Parameters

- **key** –
- **chat\_id** –
- **user\_id** –

#### Returns

**async\_task** (*func*)

Execute handler as task and return None. Use this decorator for slow handlers (with timeouts)

```
@dp.message_handler(commands=['command'])
@dp.async_task
async def cmd_with_timeout(message: types.Message):
    await asyncio.sleep(120)
    return SendMessage(message.chat.id, 'KABOOM').reply(message)
```

**Parameters** *func* –

**Returns**

**throttled** (*on\_throttled: Optional[Callable] = None, key=None, rate=None, user\_id=None, chat\_id=None*)

Meta-decorator for throttling. Invokes on\_throttled if the handler was throttled.

Example:

```
async def handler_throttled(message: types.Message, **kwargs):
    await message.answer("Throttled!")

@dp.throttled(handler_throttled)
async def some_handler(message: types.Message):
    await message.answer("Didn't throttled!")
```

**Parameters**

- **on\_throttled** – the callable object that should be either a function or return a coroutine
- **key** – key in storage
- **rate** – limit (by default is equal to default rate limit)
- **user\_id** – user id
- **chat\_id** – chat id

**Returns** decorator

**bind\_filter** (*callback: Union[Callable, aiogram.dispatcher.filters.filters.AbstractFilter], validator: Optional[Callable] = None, event\_handlers: Optional[List[aiogram.dispatcher.handler.Handler]] = None, exclude\_event\_handlers: Optional[Iterable[aiogram.dispatcher.handler.Handler]] = None*)

Register filter

**Parameters**

- **callback** – callable or subclass of AbstractFilter
- **validator** – custom validator.
- **event\_handlers** – list of instances of Handler
- **exclude\_event\_handlers** – list of excluded event handlers (Handler)

**unbind\_filter** (*callback: Union[Callable, aiogram.dispatcher.filters.filters.AbstractFilter]*)

Unregister filter

**Parameters** **callback** – callable of subclass of AbstractFilter

**setup\_middleware** (*middleware*)

Setup middleware

Parameters middleware –

Returns

## 4.6 Utils

### 4.6.1 Auth Widget

Implementation of Telegram site authorization checking mechanism for more information <https://core.telegram.org/widgets/login#checking-authorization>

Source: <https://gist.github.com/JrooTJunior/887791de7273c9df5277d2b1ecadc839>

`aiogram.utils.auth_widget.generate_hash(data: dict, token: str) → str`  
Generate secret hash

**Parameters**

- **data** –
- **token** –

**Returns**

`aiogram.utils.auth_widget.check_token(data: dict, token: str) → bool`  
Validate auth token

**Parameters**

- **data** –
- **token** –

**Returns**

`aiogram.utils.auth_widget.check_signature(token: str, hash: str, **kwargs) → bool`  
Generate hexadecimal representation of the HMAC-SHA-256 signature of the data-check-string with the SHA256 hash of the bot's token used as a secret key

**Parameters**

- **token** –
- **hash** –
- **kwargs** – all params received on auth

**Returns**

`aiogram.utils.auth_widget.check_integrity(token: str, data: dict) → bool`  
Verify the authentication and the integrity of the data received on user's auth

**Parameters**

- **token** – Bot's token
- **data** – all data that came on auth

**Returns**

## 4.6.2 Executor

```
aiogram.utils.executor.start_polling(dispatcher, *, loop=None, skip_updates=False,
                                     reset_webhook=True, on_startup=None,
                                     on_shutdown=None, timeout=20, relax=0.1,
                                     fast=True)
```

Start bot in long-polling mode

### Parameters

- **dispatcher** –
- **loop** –
- **skip\_updates** –
- **reset\_webhook** –
- **on\_startup** –
- **on\_shutdown** –
- **timeout** –

```
aiogram.utils.executor.set_webhook(dispatcher: aiogram.dispatcher.dispatcher.Dispatcher,
                                   webhook_path: str, *, loop: Optional[asyncio.events.AbstractEventLoop] = None,
                                   skip_updates: bool = None, on_startup: Optional[Callable] = None, on_shutdown: Optional[Callable] = None,
                                   check_ip: bool = False, retry_after: Optional[Union[str, int]] = None,
                                   route_name: str = 'webhook_handler', web_app: Optional[aiohttp.web_app.Application] = None)
```

Set webhook for bot

### Parameters

- **dispatcher** – Dispatcher
- **webhook\_path** – str
- **loop** – Optional[asyncio.AbstractEventLoop] (default: None)
- **skip\_updates** – bool (default: None)
- **on\_startup** – Optional[Callable] (default: None)
- **on\_shutdown** – Optional[Callable] (default: None)
- **check\_ip** – bool (default: False)
- **retry\_after** – Optional[Union[str, int]] See <https://tools.ietf.org/html/rfc7231#section-7.1.3> (default: None)
- **route\_name** – str (default: 'webhook\_handler')
- **web\_app** – Optional[Application] (default: None)

### Returns

```
aiogram.utils.executor.start_webhook(dispatcher, webhook_path, *, loop=None,
                                     skip_updates=None, on_startup=None,
                                     on_shutdown=None, check_ip=False,
                                     retry_after=None, route_name='webhook_handler',
                                     **kwargs)
```

Start bot in webhook mode

#### Parameters

- **dispatcher** –
- **webhook\_path** –
- **loop** –
- **skip\_updates** –
- **on\_startup** –
- **on\_shutdown** –
- **check\_ip** –
- **route\_name** –
- **kwargs** –

#### Returns

`aiogram.utils.executor.start(dispatcher, future, *, loop=None, skip_updates=None, on_startup=None, on_shutdown=None)`

Execute Future.

#### Parameters

- **dispatcher** – instance of Dispatcher
- **future** – future
- **loop** – instance of AbstractEventLoop
- **skip\_updates** –
- **on\_startup** –
- **on\_shutdown** –

#### Returns

`class aiogram.utils.executor.Executor(dispatcher, skip_updates=None, check_ip=False, retry_after=None, loop=None)`

Main executor class

**set\_web\_app** (*application: aiohttp.web\_app.Application*)

Change instance of aiohttp.web.Applicaton

#### Parameters application –

**on\_startup** (*callback: callable, polling=True, webhook=True*)

Register a callback for the startup process

#### Parameters

- **callback** –
- **polling** – use with polling
- **webhook** – use with webhook

**on\_shutdown** (*callback: callable, polling=True, webhook=True*)

Register a callback for the shutdown process

#### Parameters

- **callback** –

- **polling** – use with polling
- **webhook** – use with webhook

**set\_webhook** (*webhook\_path: Optional[str] = None, request\_handler: Any = <class 'aiogram.dispatcher.webhook.WebhookRequestHandler'>, route\_name: str = 'webhook\_handler', web\_app: Optional[aiohttp.web\_app.Application] = None*)

Set webhook for bot

#### Parameters

- **webhook\_path** – Optional[str] (default: None)
- **request\_handler** – Any (default: WebhookRequestHandler)
- **route\_name** – str Name of webhook handler route (default: 'webhook\_handler')
- **web\_app** – Optional[Application] (default: None)

#### Returns

**start\_webhook** (*webhook\_path=None, request\_handler=<class 'aiogram.dispatcher.webhook.WebhookRequestHandler'>, route\_name='webhook\_handler', \*\*kwargs*)

Start bot in webhook mode

#### Parameters

- **webhook\_path** –
- **request\_handler** –
- **route\_name** – Name of webhook handler route
- **kwargs** –

#### Returns

**start\_polling** (*reset\_webhook=None, timeout=20, relax=0.1, fast=True*)

Start bot in long-polling mode

#### Parameters

- **reset\_webhook** –
- **timeout** –

**start** (*future*)

Execute Future.

Return the Future's result, or raise its exception.

#### Parameters **future** –

#### Returns

### 4.6.3 Exceptions

- **TelegramAPIError**
  - **ValidationError**
  - **Throttled**
  - **BadRequest**
  - \* **MessageError**



- MessageNotModified
- MessageToForwardNotFound
- MessageToDeleteNotFound
- MessageToPinNotFound
- MessageIdentifierNotSpecified
- MessageTextIsEmpty
- MessageCantBeEdited
- MessageCantBeDeleted
- MessageCantBeForwarded
- MessageToEditNotFound
- MessageToReplyNotFound
- ToMuchMessages
- \* **PollError**
  - PollCantBeStopped
  - PollHasAlreadyClosed
  - PollsCantBeSentToPrivateChats
  - **PollSizeError**
    - PollMustHaveMoreOptions
    - PollCantHaveMoreOptions
    - PollsOptionsLengthTooLong
    - PollOptionsMustBeNonEmpty
    - PollQuestionMustBeNonEmpty
  - MessageWithPollNotFound (with MessageError)
  - MessageIsNotAPoll (with MessageError)
- \* ObjectExpectedAsReplyMarkup
- \* InlineKeyboardExpected
- \* ChatNotFound
- \* ChatDescriptionIsNotModified
- \* InvalidQueryID
- \* InvalidPeerID
- \* InvalidHTTPUrlContent
- \* ButtonURLInvalid
- \* URLHostIsEmpty
- \* StartParamInvalid
- \* ButtonDataInvalid
- \* FileIsTooBig

- \* WrongFileIdentifier
- \* GroupDeactivated
- \* **BadWebhook**
  - WebhookRequireHTTPS
  - BadWebhookPort
  - BadWebhookAddrInfo
  - BadWebhookNoAddressAssociatedWithHostname
- \* **NotFound**
  - MethodNotKnown
- \* PhotoAsInputFileRequired
- \* InvalidStickersSet
- \* NoStickerInRequest
- \* ChatAdminRequired
- \* NeedAdministratorRightsInTheChannel
- \* MethodNotAvailableInPrivateChats
- \* CantDemoteChatCreator
- \* CantRestrictSelf
- \* NotEnoughRightsToRestrict
- \* PhotoDimensions
- \* UnavailableMembers
- \* TypeOfFileMismatch
- \* WrongRemoteFileIdSpecified
- \* PaymentProviderInvalid
- \* CurrencyTotalAmountInvalid
- \* CantParseUrl
- \* UnsupportedUrlProtocol
- \* CantParseEntities
- \* ResultIdDuplicate
- \* MethodIsNotAvailable
- **ConflictError**
  - \* TerminatedByOtherGetUpdates
  - \* CantGetUpdates
- **Unauthorized**
  - \* BotKicked
  - \* BotBlocked
  - \* UserDeactivated

- \* CantInitiateConversation
- \* CantTalkWithBots
- NetworkError
- RetryAfter
- MigrateToChat
- RestartingTelegram
- AIOGramWarning
  - TimeoutWarning

**exception** aiogram.utils.exceptions.TelegramAPIError (message=None)

**exception** aiogram.utils.exceptions.AIOGramWarning

**exception** aiogram.utils.exceptions.TimeoutWarning

**exception** aiogram.utils.exceptions.FSMStorageWarning

**exception** aiogram.utils.exceptions.ValidationError (message=None)

**exception** aiogram.utils.exceptions.BadRequest (message=None)

**exception** aiogram.utils.exceptions.MessageError (message=None)

**exception** aiogram.utils.exceptions.MessageNotModified (message=None)

Will be raised when you try to set new text is equals to current text.

**exception** aiogram.utils.exceptions.MessageToForwardNotFound (message=None)

Will be raised when you try to forward very old or deleted or unknown message.

**exception** aiogram.utils.exceptions.MessageToDeleteNotFound (message=None)

Will be raised when you try to delete very old or deleted or unknown message.

**exception** aiogram.utils.exceptions.MessageToPinNotFound (message=None)

Will be raised when you try to pin deleted or unknown message.

**exception** aiogram.utils.exceptions.MessageToReplyNotFound (message=None)

Will be raised when you try to reply to very old or deleted or unknown message.

**exception** aiogram.utils.exceptions.MessageIdentifierNotSpecified (message=None)

**exception** aiogram.utils.exceptions.MessageTextIsEmpty (message=None)

**exception** aiogram.utils.exceptions.MessageCantBeEdited (message=None)

**exception** aiogram.utils.exceptions.MessageCantBeDeleted (message=None)

**exception** aiogram.utils.exceptions.MessageCantBeForwarded (message=None)

**exception** aiogram.utils.exceptions.MessageToEditNotFound (message=None)

**exception** aiogram.utils.exceptions.MessageIsTooLong (message=None)

**exception** aiogram.utils.exceptions.ToMuchMessages (message=None)

Will be raised when you try to send media group with more than 10 items.

**exception** aiogram.utils.exceptions.ObjectExpectedAsReplyMarkup (message=None)

**exception** aiogram.utils.exceptions.InlineKeyboardExpected (message=None)

**exception** aiogram.utils.exceptions.PollError (message=None)

**exception** aiogram.utils.exceptions.PollCantBeStopped (message=None)

**exception** aiogram.utils.exceptions.**PollHasAlreadyBeenClosed** (*message=None*)

**exception** aiogram.utils.exceptions.**PollsCantBeSentToPrivateChats** (*message=None*)

**exception** aiogram.utils.exceptions.**PollSizeError** (*message=None*)

**exception** aiogram.utils.exceptions.**PollMustHaveMoreOptions** (*message=None*)

**exception** aiogram.utils.exceptions.**PollCantHaveMoreOptions** (*message=None*)

**exception** aiogram.utils.exceptions.**PollOptionsMustBeNonEmpty** (*message=None*)

**exception** aiogram.utils.exceptions.**PollQuestionMustBeNonEmpty** (*message=None*)

**exception** aiogram.utils.exceptions.**PollOptionsLengthTooLong** (*message=None*)

**exception** aiogram.utils.exceptions.**PollQuestionLengthTooLong** (*message=None*)

**exception** aiogram.utils.exceptions.**PollCanBeRequestedInPrivateChatsOnly** (*message=None*)

**exception** aiogram.utils.exceptions.**MessageWithPollNotFound** (*message=None*)  
Will be raised when you try to stop poll with message without poll

**exception** aiogram.utils.exceptions.**MessageIsNotAPoll** (*message=None*)  
Will be raised when you try to stop poll with message without poll

**exception** aiogram.utils.exceptions.**ChatNotFound** (*message=None*)

**exception** aiogram.utils.exceptions.**ChatIdIsEmpty** (*message=None*)

**exception** aiogram.utils.exceptions.**InvalidUserId** (*message=None*)

**exception** aiogram.utils.exceptions.**ChatDescriptionIsNotModified** (*message=None*)

**exception** aiogram.utils.exceptions.**InvalidQueryID** (*message=None*)

**exception** aiogram.utils.exceptions.**InvalidPeerID** (*message=None*)

**exception** aiogram.utils.exceptions.**InvalidHttpRequestContent** (*message=None*)

**exception** aiogram.utils.exceptions.**ButtonURLInvalid** (*message=None*)

**exception** aiogram.utils.exceptions.**URLHostIsEmpty** (*message=None*)

**exception** aiogram.utils.exceptions.**StartParamInvalid** (*message=None*)

**exception** aiogram.utils.exceptions.**ButtonDataInvalid** (*message=None*)

**exception** aiogram.utils.exceptions.**FileIsTooBig** (*message=None*)

**exception** aiogram.utils.exceptions.**WrongFileIdentifier** (*message=None*)

**exception** aiogram.utils.exceptions.**GroupDeactivated** (*message=None*)

**exception** aiogram.utils.exceptions.**PhotoAsInputFileRequired** (*message=None*)  
Will be raised when you try to set chat photo from file ID.

**exception** aiogram.utils.exceptions.**InvalidStickersSet** (*message=None*)

**exception** aiogram.utils.exceptions.**NoStickerInRequest** (*message=None*)

**exception** aiogram.utils.exceptions.**ChatAdminRequired** (*message=None*)

**exception** aiogram.utils.exceptions.**NeedAdministratorRightsInTheChannel** (*message=None*)

**exception** aiogram.utils.exceptions.**NotEnoughRightsToPinMessage** (*message=None*)

**exception** aiogram.utils.exceptions.**MethodNotAvailableInPrivateChats** (*message=None*)

**exception** aiogram.utils.exceptions.**CantDemoteChatCreator** (*message=None*)

---

```
exception aiogram.utils.exceptions.CantRestrictSelf (message=None)
exception aiogram.utils.exceptions.NotEnoughRightsToRestrict (message=None)
exception aiogram.utils.exceptions.PhotoDimensions (message=None)
exception aiogram.utils.exceptions.UnavailableMembers (message=None)
exception aiogram.utils.exceptions.TypeOfFileMismatch (message=None)
exception aiogram.utils.exceptions.WrongRemoteFileIdSpecified (message=None)
exception aiogram.utils.exceptions.PaymentProviderInvalid (message=None)
exception aiogram.utils.exceptions.CurrencyTotalAmountInvalid (message=None)
exception aiogram.utils.exceptions.BadWebhook (message=None)
exception aiogram.utils.exceptions.WebhookRequireHTTPS (message=None)
exception aiogram.utils.exceptions.BadWebhookPort (message=None)
exception aiogram.utils.exceptions.BadWebhookAddrInfo (message=None)
exception aiogram.utils.exceptions.BadWebhookNoAddressAssociatedWithHostname (message=None)
exception aiogram.utils.exceptions.CantParseUrl (message=None)
exception aiogram.utils.exceptions.UnsupportedUrlProtocol (message=None)
exception aiogram.utils.exceptions.CantParseEntities (message=None)
exception aiogram.utils.exceptions.ResultIdDuplicate (message=None)
exception aiogram.utils.exceptions.BotDomainInvalid (message=None)
exception aiogram.utils.exceptions.MethodIsNotAvailable (message=None)
exception aiogram.utils.exceptions.NotFound (message=None)
exception aiogram.utils.exceptions.MethodNotKnown (message=None)
exception aiogram.utils.exceptions.ConflictError (message=None)
exception aiogram.utils.exceptions.TerminatedByOtherGetUpdates (message=None)
exception aiogram.utils.exceptions.CantGetUpdates (message=None)
exception aiogram.utils.exceptions.Unauthorized (message=None)
exception aiogram.utils.exceptions.BotKicked (message=None)
exception aiogram.utils.exceptions.BotBlocked (message=None)
exception aiogram.utils.exceptions.UserDeactivated (message=None)
exception aiogram.utils.exceptions.CantInitiateConversation (message=None)
exception aiogram.utils.exceptions.CantTalkWithBots (message=None)
exception aiogram.utils.exceptions.NetworkError (message=None)
exception aiogram.utils.exceptions.RestartingTelegram
exception aiogram.utils.exceptions.RetryAfter (retry_after)
exception aiogram.utils.exceptions.MigrateToChat (chat_id)
exception aiogram.utils.exceptions.Throttled (**kwargs)
```

## 4.6.4 Markdown

`aiogram.utils.markdown.quote_html(*content, sep='') → str`

Quote HTML symbols

All `<`, `>`, `&` and `"` symbols that are not a part of a tag or an HTML entity must be replaced with the corresponding HTML entities (`<` with `&lt;`; `>` with `&gt;`; `&` with `&amp` and `"` with `&quot;`).

**Parameters**

- **content** –
- **sep** –

**Returns**

`aiogram.utils.markdown.escape_md(*content, sep='') → str`

Escape markdown text

E.g. for usernames

**Parameters**

- **content** –
- **sep** –

**Returns**

`aiogram.utils.markdown.text(*content, sep='')`

Join all elements with a separator

**Parameters**

- **content** –
- **sep** –

**Returns**

`aiogram.utils.markdown.bold(*content, sep='') → str`

Make bold text (Markdown)

**Parameters**

- **content** –
- **sep** –

**Returns**

`aiogram.utils.markdown.hbold(*content, sep='') → str`

Make bold text (HTML)

**Parameters**

- **content** –
- **sep** –

**Returns**

`aiogram.utils.markdown.italic(*content, sep='') → str`

Make italic text (Markdown)

**Parameters**

- **content** –
- **sep** –

**Returns**

`aiogram.utils.markdown.hitalic(*content, sep='') → str`

Make italic text (HTML)

**Parameters**

- **content** –

- **sep** –

**Returns**

`aiogram.utils.markdown.code(*content, sep='') → str`

Make mono-width text (Markdown)

**Parameters**

- **content** –

- **sep** –

**Returns**

`aiogram.utils.markdown.hcode(*content, sep='') → str`

Make mono-width text (HTML)

**Parameters**

- **content** –

- **sep** –

**Returns**

`aiogram.utils.markdown.pre(*content, sep='\n') → str`

Make mono-width text block (Markdown)

**Parameters**

- **content** –

- **sep** –

**Returns**

`aiogram.utils.markdown.hpre(*content, sep='\n') → str`

Make mono-width text block (HTML)

**Parameters**

- **content** –

- **sep** –

**Returns**

`aiogram.utils.markdown.underline(*content, sep='') → str`

Make underlined text (Markdown)

**Parameters**

- **content** –

- **sep** –

**Returns**

`aiogram.utils.markdown.hunderline(*content, sep='') → str`

Make underlined text (HTML)

**Parameters**

- **content** –

- **sep** –

**Returns**

`aiogram.utils.markdown.strikethrough(*content, sep='') → str`

Make strikethrough text (Markdown)

**Parameters**

- **content** –

- **sep** –

**Returns**

`aiogram.utils.markdown.hstrikethrough(*content, sep='') → str`

Make strikethrough text (HTML)

**Parameters**

- **content** –
- **sep** –

**Returns**

`aiogram.utils.markdown.link(title: str, url: str) → str`

Format URL (Markdown)

**Parameters**

- **title** –
- **url** –

**Returns**

`aiogram.utils.markdown.hlink(title: str, url: str) → str`

Format URL (HTML)

**Parameters**

- **title** –
- **url** –

**Returns**

`aiogram.utils.markdown.hide_link(url: str) → str`

Hide URL (HTML only) Can be used for adding an image to a text message

**Parameters** **url** –**Returns**

## 4.6.5 Helper

**Example:**

```
>>> from aiogram.utils.helper import Helper, ListItem, HelperMode, Item
>>> class MyHelper(Helper):
...     mode = HelperMode.lowerCamelCase
...     FOO_ITEM = ListItem()
...     BAR_ITEM = ListItem()
...     BAZ_ITEM = ListItem()
...     LOREM = Item()
...
>>> print(MyHelper.FOO_ITEM & MyHelper.BAR_ITEM)
<<< ['fooItem', 'barItem']
>>> print(MyHelper.all())
<<< ['barItem', 'bazItem', 'fooItem', 'lorem']
```

**class** `aiogram.utils.helper.Item(value=None)`

Helper item

If a value is not provided, it will be automatically generated based on a variable's name

**class** `aiogram.utils.helper.ListItem(value=None)`

This item is always a list

You can use `&`, `|` and `+` operators for that.



```
class aiogram.utils.helper.ItemsList(*seq)
```

Patch for default list

This class provides +, &, !, +=, &=, != operators for extending the list

## 4.6.6 Deprecated

aiogram.utils.deprecated.**deprecated**(reason, stacklevel=2) → Callable

This is a decorator which can be used to mark functions as deprecated. It will result in a warning being emitted when the function is used.

Source: <https://stackoverflow.com/questions/2536307/decorators-in-the-python-standard-lib-deprecated-specifically>

```
aiogram.utils.deprecated.renamed_argument(old_name: str, new_name: str, until_version:
                                          str, stacklevel: int = 3)
```

A meta-decorator to mark an argument as deprecated.

```
@renamed_argument("chat", "chat_id", "3.0") # stacklevel=3 by default
@renamed_argument("user", "user_id", "3.0", stacklevel=4)
def some_function(user_id, chat_id=None):
    print(f"user_id={user_id}, chat_id={chat_id}")

some_function(user=123) # prints 'user_id=123, chat_id=None' with warning
some_function(123) # prints 'user_id=123, chat_id=None' without warning
some_function(user_id=123) # prints 'user_id=123, chat_id=None' without warning
```

### Parameters

- **old\_name** –
- **new\_name** –
- **until\_version** – the version in which the argument is scheduled to be removed
- **stacklevel** – leave it to default if it's the first decorator used.

Increment with any new decorator used. :return: decorator

```
class aiogram.utils.deprecated.DeprecatedReadOnlyClassVar(warning_message:
                                                         str, new_value_getter:
                                                         Callable[[_OwnerCls],
                                                         _VT])
```

DeprecatedReadOnlyClassVar[Owner, ValueType]

### Parameters

- **warning\_message** – Warning message when getter gets called
- **new\_value\_getter** – Any callable with (owner\_class: Type[Owner]) -> Value-  
Type signature that will be executed

Usage example:

```
>>> class MyClass:
...     some_attribute: DeprecatedReadOnlyClassVar[MyClass, int] = ...
↳     DeprecatedReadOnlyClassVar(
...         "Warning message.", lambda owner: 15)
...
>>> MyClass.some_attribute # does warning.warn with `Warning message` and
↳ returns 15 in the current case
```

### 4.6.7 Payload

`aiogram.utils.payload.generate_payload(exclude=None, **kwargs)`

Generate payload

Usage: `payload = generate_payload(**locals(), exclude=['foo'])`

**Parameters**

- **exclude** –
- **kwargs** –

**Returns** dict

`aiogram.utils.payload.prepare_arg(value)`

Stringify dicts/lists and convert datetime/timedelta to unix-time

**Parameters** **value** –

**Returns**

### 4.6.8 Parts

`aiogram.utils.parts.split_text(text: str, length: int = 4096) → List[str]`

Split long text

**Parameters**

- **text** –
- **length** –

**Returns** list of parts

**Return type** `typing.List[str]`

`aiogram.utils.parts.safe_split_text(text: str, length: int = 4096) → List[str]`

Split long text

**Parameters**

- **text** –
- **length** –

**Returns**

`aiogram.utils.parts.paginate(data: Iterable, page: int = 0, limit: int = 10) → Iterable`

Slice data over pages

**Parameters**

- **data** (`typing.Iterable`) – any iterable object
- **page** (`int`) – number of page
- **limit** (`int`) – items per page

**Returns** sliced object

**Return type** `typing.Iterable`

## 4.6.9 JSON

## 4.6.10 Emoji

## 4.6.11 Deep linking

Deep linking

Telegram bots have a deep linking mechanism, that allows for passing additional parameters to the bot on startup. It could be a command that launches the bot — or an auth token to connect the user's Telegram account to their account on some external service.

You can read detailed description in the source: <https://core.telegram.org/bots#deep-linking>

We have add some utils to get deep links more handy.

Basic link example:

```
from aiogram.utils.deep_linking import get_start_link
link = await get_start_link('foo') # result: 'https://t.me/MyBot?start=foo'
```

Encoded link example:

```
from aiogram.utils.deep_linking import get_start_link, decode_payload
link = await get_start_link('foo', encode=True) # result: 'https://t.me/
↳MyBot?start=Zm9v'
# and decode it back:
payload = decode_payload('Zm9v') # result: 'foo'
```

**async** `aiogram.utils.deep_linking.get_start_link(payload: str, encode=False) → str`

Use this method to handy get 'start' deep link with your payload. If you need to encode payload or pass special characters - set encode as True

**Parameters**

- **payload** – args passed with /start
- **encode** – encode payload with base64url

**Returns** link

**async** `aiogram.utils.deep_linking.get_startgroup_link(payload: str, encode=False) → str`

Use this method to handy get 'startgroup' deep link with your payload. If you need to encode payload or pass special characters - set encode as True

**Parameters**

- **payload** – args passed with /start
- **encode** – encode payload with base64url

**Returns** link

`aiogram.utils.deep_linking.encode_payload(payload: str) → str`  
Encode payload with URL-safe base64url.

`aiogram.utils.deep_linking.decode_payload(payload: str) → str`  
Decode payload with URL-safe base64url.

`aiogram.utils.deep_linking.filter_payload(payload: str) → str`  
Convert payload to text and search for not allowed symbols.

## 4.7 Examples

### 4.7.1 Echo bot

Listing 1: echo\_bot.py

```
1  """
2  This is a echo bot.
3  It echoes any incoming text messages.
4  """
5
6  import logging
7
8  from aiogram import Bot, Dispatcher, executor, types
9
10 API_TOKEN = 'BOT TOKEN HERE'
11
12 # Configure logging
13 logging.basicConfig(level=logging.INFO)
14
15 # Initialize bot and dispatcher
16 bot = Bot(token=API_TOKEN)
17 dp = Dispatcher(bot)
18
19
20 @dp.message_handler(commands=['start', 'help'])
21 async def send_welcome(message: types.Message):
22     """
23     This handler will be called when user sends `/start` or `/help` command
24     """
25     await message.reply("Hi!\nI'm EchoBot!\nPowered by aiogram.")
26
27
28 @dp.message_handler(regexp='^cat[s]?$|puss)')
29 async def cats(message: types.Message):
30     with open('data/cats.jpg', 'rb') as photo:
31         '''
32         # Old fashioned way:
33         await bot.send_photo(
34             message.chat.id,
35             photo,
36             caption='Cats are here ',
37             reply_to_message_id=message.message_id,
38         )
39         '''
40
41     await message.reply_photo(photo, caption='Cats are here ')
42
43
44 @dp.message_handler()
45 async def echo(message: types.Message):
46     # old style:
47     # await bot.send_message(message.chat.id, message.text)
48
49     await message.answer(message.text)
50
```

(continues on next page)

(continued from previous page)

```

51
52 if __name__ == '__main__':
53     executor.start_polling(dp, skip_updates=True)

```

## 4.7.2 Inline bot

Listing 2: inline\_bot.py

```

1  import hashlib
2  import logging
3
4  from aiogram import Bot, Dispatcher, executor
5  from aiogram.types import InlineQuery, \
6      InputTextMessageContent, InlineQueryResultArticle
7
8  API_TOKEN = 'BOT_TOKEN_HERE'
9
10 logging.basicConfig(level=logging.DEBUG)
11
12 bot = Bot(token=API_TOKEN)
13 dp = Dispatcher(bot)
14
15
16 @dp.inline_handler()
17 async def inline_echo(inline_query: InlineQuery):
18     # id affects both preview and content,
19     # so it has to be unique for each result
20     # (Unique identifier for this result, 1-64 Bytes)
21     # you can set your unique id's
22     # but for example i'll generate it based on text because I know, that
23     # only text will be passed in this example
24     text = inline_query.query or 'echo'
25     input_content = InputTextMessageContent(text)
26     result_id: str = hashlib.md5(text.encode()).hexdigest()
27     item = InlineQueryResultArticle(
28         id=result_id,
29         title=f'Result {text!r}',
30         input_message_content=input_content,
31     )
32     # don't forget to set cache_time=1 for testing (default is 300s or 5m)
33     await bot.answer_inline_query(inline_query.id, results=[item], cache_time=1)
34
35
36 if __name__ == '__main__':
37     executor.start_polling(dp, skip_updates=True)

```

### 4.7.3 Advanced executor example

Listing 3: advanced\_executor\_example.py

```

1  #!/usr/bin/env python3
2  """
3  **This example is outdated**
4  In this example used ArgumentParser for configuring Your bot.
5
6  Provided to start bot with webhook:
7      python advanced_executor_example.py \
8          --token TOKEN_HERE \
9          --host 0.0.0.0 \
10         --port 8084 \
11         --host-name example.com \
12         --webhook-port 443
13
14  Or long polling:
15      python advanced_executor_example.py --token TOKEN_HERE
16
17  So... In this example found small trouble:
18      can't get bot instance in handlers.
19
20
21  If you want to automatic change getting updates method use executor utils (from_
22  ↪ aiogram.utils.executor)
23  """
24  # TODO: Move token to environment variables.
25
26  import argparse
27  import logging
28  import ssl
29  import sys
30
31  from aiogram import Bot
32  from aiogram.dispatcher import Dispatcher
33  from aiogram.dispatcher.webhook import *
34  from aiogram.utils.executor import start_polling, start_webhook
35
36  logging.basicConfig(level=logging.INFO)
37
38  # Configure arguments parser.
39  parser = argparse.ArgumentParser(description='Python telegram bot')
40  parser.add_argument('--token', '-t', nargs='?', type=str, default=None, help='Set_
41  ↪ working directory')
42  parser.add_argument('--sock', help='UNIX Socket path')
43  parser.add_argument('--host', help='Webserver host')
44  parser.add_argument('--port', type=int, help='Webserver port')
45  parser.add_argument('--cert', help='Path to SSL certificate')
46  parser.add_argument('--pkey', help='Path to SSL private key')
47  parser.add_argument('--host-name', help='Set webhook host name')
48  parser.add_argument('--webhook-port', type=int, help='Port for webhook (default=port)
49  ↪')
50  parser.add_argument('--webhook-path', default='/webhook', help='Port for webhook_
51  ↪ (default=port)')

```

(continues on next page)

(continued from previous page)

```

50 async def cmd_start(message: types.Message):
51     return SendMessage(message.chat.id, f"Hello, {message.from_user.full_name}!")
52
53
54 def setup_handlers(dispatcher: Dispatcher):
55     # This example has only one messages handler
56     dispatcher.register_message_handler(cmd_start, commands=['start', 'welcome'])
57
58
59 async def on_startup(dispatcher, url=None, cert=None):
60     setup_handlers(dispatcher)
61
62     bot = dispatcher.bot
63
64     # Get current webhook status
65     webhook = await bot.get_webhook_info()
66
67     if url:
68         # If URL is bad
69         if webhook.url != url:
70             # If URL doesnt match with by current remove webhook
71             if not webhook.url:
72                 await bot.delete_webhook()
73
74             # Set new URL for webhook
75             if cert:
76                 with open(cert, 'rb') as cert_file:
77                     await bot.set_webhook(url, certificate=cert_file)
78             else:
79                 await bot.set_webhook(url)
80     elif webhook.url:
81         # Otherwise remove webhook.
82         await bot.delete_webhook()
83
84
85 async def on_shutdown(dispatcher):
86     print('Shutdown.')
87
88
89 def main(arguments):
90     args = parser.parse_args(arguments)
91     token = args.token
92     sock = args.sock
93     host = args.host
94     port = args.port
95     cert = args.cert
96     pkey = args.pkey
97     host_name = args.host_name or host
98     webhook_port = args.webhook_port or port
99     webhook_path = args.webhook_path
100
101     # Fi webhook path
102     if not webhook_path.startswith('/'):
103         webhook_path = '/' + webhook_path
104
105     # Generate webhook URL
106     webhook_url = f"https://{host_name}:{webhook_port}{webhook_path}"

```

(continues on next page)

(continued from previous page)

```

107
108     # Create bot & dispatcher instances.
109     bot = Bot(token)
110     dispatcher = Dispatcher(bot)
111
112     if (sock or host) and host_name:
113         if cert and pkey:
114             ssl_context = ssl.SSLContext(ssl.PROTOCOL_TLSv1_2)
115             ssl_context.load_cert_chain(cert, pkey)
116         else:
117             ssl_context = None
118
119     start_webhook(dispatcher, webhook_path,
120                  on_startup=functools.partial(on_startup, url=webhook_url,
121 ↪ cert=cert),
122                  on_shutdown=on_shutdown,
123                  host=host, port=port, path=sock, ssl_context=ssl_context)
124
125     else:
126         start_polling(dispatcher, on_startup=on_startup, on_shutdown=on_shutdown)
127
128 if __name__ == '__main__':
129     argv = sys.argv[1:]
130
131     if not len(argv):
132         parser.print_help()
133         sys.exit(1)
134
135     main(argv)

```

## 4.7.4 Proxy and emoji

Listing 4: proxy\_and\_emoji.py

```

1  import logging
2
3  import aiohttp
4
5  from aiogram import Bot, types
6  from aiogram.dispatcher import Dispatcher
7  from aiogram.types import ParseMode
8  from aiogram.utils.emoji import emojiize
9  from aiogram.utils.executor import start_polling
10 from aiogram.utils.markdown import bold, code, italic, text
11
12 # Configure bot here
13 API_TOKEN = 'BOT_TOKEN_HERE'
14 PROXY_URL = 'http://PROXY_URL' # Or 'socks5://host:port'
15
16 # NOTE: If authentication is required in your proxy then uncomment next line and
17 ↪ change login/password for it
18 # PROXY_AUTH = aiohttp.BasicAuth(login='login', password='password')
19 # And add `proxy_auth=PROXY_AUTH` argument in line 30, like this:
20 # >>> bot = Bot(token=API_TOKEN, proxy=PROXY_URL, proxy_auth=PROXY_AUTH)
21 # Also you can use Socks5 proxy but you need manually install aiohttp_socks package.

```

(continues on next page)



(continued from previous page)

```

21
22 # Get my ip URL
23 GET_IP_URL = 'http://bot.whatismyipaddress.com/'
24
25 logging.basicConfig(level=logging.INFO)
26
27 bot = Bot(token=API_TOKEN, proxy=PROXY_URL)
28
29 # If auth is required:
30 # bot = Bot(token=API_TOKEN, proxy=PROXY_URL, proxy_auth=PROXY_AUTH)
31 dp = Dispatcher(bot)
32
33
34 async def fetch(url, session):
35     async with session.get(url) as response:
36         return await response.text()
37
38
39 @dp.message_handler(commands=['start'])
40 async def cmd_start(message: types.Message):
41     # fetching urls will take some time, so notify user that everything is OK
42     await types.ChatActions.typing()
43
44     content = []
45
46     # Make request (without proxy)
47     async with aiohttp.ClientSession() as session:
48         ip = await fetch(GET_IP_URL, session)
49         content.append(text(':globe_showing_Americas:', bold('IP:'), code(ip)))
50         # This line is formatted to ' *IP:* `YOUR IP` '
51
52     # Make request through bot's proxy
53     ip = await fetch(GET_IP_URL, bot.session)
54     content.append(text(':locked_with_key:', bold('IP:'), code(ip), italic('via proxy
↳ '))))
55     # This line is formatted to ' *IP:* `YOUR IP` _via proxy_'
56
57     # Send content
58     await bot.send_message(message.chat.id, emojiize(text(*content, sep='\n')), parse_
↳ mode=ParseMode.MARKDOWN)
59
60     # In this example you can see emoji codes: ":globe_showing_Americas:" and
↳ ":locked_with_key:"
61     # You can find full emoji cheat sheet at https://www.webpagefx.com/tools/emoji-
↳ cheat-sheet/
62     # For representing emoji codes into real emoji use emoji util (aiogram.utils.
↳ emoji)
63     # (you have to install emoji module)
64
65     # For example emojiize('Moon face :new_moon_face:') is transformed to 'Moon face '
66
67
68 if __name__ == '__main__':
69     start_polling(dp, skip_updates=True)

```

## 4.7.5 Finite state machine example

Listing 5: finite\_state\_machine\_example.py

```

1  import logging
2
3  import aiogram.utils.markdown as md
4  from aiogram import Bot, Dispatcher, types
5  from aiogram.contrib.fsm_storage.memory import MemoryStorage
6  from aiogram.dispatcher import FSMContext
7  from aiogram.dispatcher.filters import Text
8  from aiogram.dispatcher.filters.state import State, StatesGroup
9  from aiogram.types import ParseMode
10 from aiogram.utils import executor
11
12 logging.basicConfig(level=logging.INFO)
13
14 API_TOKEN = 'BOT TOKEN HERE'
15
16
17 bot = Bot(token=API_TOKEN)
18
19 # For example use simple MemoryStorage for Dispatcher.
20 storage = MemoryStorage()
21 dp = Dispatcher(bot, storage=storage)
22
23
24 # States
25 class Form(StatesGroup):
26     name = State() # Will be represented in storage as 'Form:name'
27     age = State() # Will be represented in storage as 'Form:age'
28     gender = State() # Will be represented in storage as 'Form:gender'
29
30
31 @dp.message_handler(commands='start')
32 async def cmd_start(message: types.Message):
33     """
34     Conversation's entry point
35     """
36     # Set state
37     await Form.name.set()
38
39     await message.reply("Hi there! What's your name?")
40
41
42 # You can use state '*' if you need to handle all states
43 @dp.message_handler(state='*', commands='cancel')
44 @dp.message_handler(Text(equals='cancel', ignore_case=True), state='*')
45 async def cancel_handler(message: types.Message, state: FSMContext):
46     """
47     Allow user to cancel any action
48     """
49     current_state = await state.get_state()
50     if current_state is None:
51         return
52
53     logging.info('Cancelling state %r', current_state)

```

(continues on next page)

(continued from previous page)

```

54     # Cancel state and inform user about it
55     await state.finish()
56     # And remove keyboard (just in case)
57     await message.reply('Cancelled.', reply_markup=types.ReplyKeyboardRemove())
58
59
60 @dp.message_handler(state=Form.name)
61 async def process_name(message: types.Message, state: FSMContext):
62     """
63     Process user name
64     """
65     async with state.proxy() as data:
66         data['name'] = message.text
67
68     await Form.next()
69     await message.reply("How old are you?")
70
71
72 # Check age. Age gotta be digit
73 @dp.message_handler(lambda message: not message.text.isdigit(), state=Form.age)
74 async def process_age_invalid(message: types.Message):
75     """
76     If age is invalid
77     """
78     return await message.reply("Age gotta be a number.\nHow old are you? (digits only)
79     ↪")
80
81
82 @dp.message_handler(lambda message: message.text.isdigit(), state=Form.age)
83 async def process_age(message: types.Message, state: FSMContext):
84     # Update state and data
85     await Form.next()
86     await state.update_data(age=int(message.text))
87
88     # Configure ReplyKeyboardMarkup
89     markup = types.ReplyKeyboardMarkup(resize_keyboard=True, selective=True)
90     markup.add("Male", "Female")
91     markup.add("Other")
92
93     await message.reply("What is your gender?", reply_markup=markup)
94
95 @dp.message_handler(lambda message: message.text not in ["Male", "Female", "Other"],
96     ↪state=Form.gender)
97 async def process_gender_invalid(message: types.Message):
98     """
99     In this example gender has to be one of: Male, Female, Other.
100    """
101    return await message.reply("Bad gender name. Choose your gender from the keyboard.
102    ↪")
103
104 @dp.message_handler(state=Form.gender)
105 async def process_gender(message: types.Message, state: FSMContext):
106     async with state.proxy() as data:
107         data['gender'] = message.text

```

(continues on next page)

(continued from previous page)

```

108     # Remove keyboard
109     markup = types.ReplyKeyboardRemove()
110
111     # And send message
112     await bot.send_message(
113         message.chat.id,
114         md.text(
115             md.text('Hi! Nice to meet you,', md.bold(data['name'])),
116             md.text('Age:', md.code(data['age'])),
117             md.text('Gender:', data['gender']),
118             sep='\n',
119         ),
120         reply_markup=markup,
121         parse_mode=ParseMode.MARKDOWN,
122     )
123
124     # Finish conversation
125     await state.finish()
126
127
128 if __name__ == '__main__':
129     executor.start_polling(dp, skip_updates=True)

```

## 4.7.6 Throttling example

Listing 6: throttling\_example.py

```

1  """
2  Example for throttling manager.
3
4  You can use that for flood controlling.
5  """
6
7  import logging
8
9  from aiogram import Bot, types
10 from aiogram.contrib.fsm_storage.memory import MemoryStorage
11 from aiogram.dispatcher import Dispatcher
12 from aiogram.utils.exceptions import Throttled
13 from aiogram.utils.executor import start_polling
14
15
16 API_TOKEN = 'BOT_TOKEN_HERE'
17
18 logging.basicConfig(level=logging.INFO)
19
20 bot = Bot(token=API_TOKEN)
21
22 # Throttling manager does not work without Leaky Bucket.
23 # You need to use a storage. For example use simple in-memory storage.
24 storage = MemoryStorage()
25 dp = Dispatcher(bot, storage=storage)
26
27
28 @dp.message_handler(commands=['start'])

```

(continues on next page)

(continued from previous page)

```

29 async def send_welcome(message: types.Message):
30     try:
31         # Execute throttling manager with rate-limit equal to 2 seconds for key "start
↪ "
32         await dp.throttle('start', rate=2)
33     except Throttled:
34         # If request is throttled, the `Throttled` exception will be raised
35         await message.reply('Too many requests!')
36     else:
37         # Otherwise do something
38         await message.reply("Hi!\nI'm EchoBot!\nPowered by aiogram.")
39
40
41 @dp.message_handler(commands=['hi'])
42 @dp.throttled(lambda msg, loop, *args, **kwargs: loop.create_task(bot.send_
↪message(msg.from_user.id, "Throttled")),
43             rate=5)
44 # loop is added to the function to run coroutines from it
45 async def say_hi(message: types.Message):
46     await message.answer("Hi")
47
48
49 # the on_throttled object can be either a regular function or coroutine
50 async def hello_throttled(*args, **kwargs):
51     # args will be the same as in the original handler
52     # kwargs will be updated with parameters given to .throttled (rate, key, user_id, ↪
↪ chat_id)
53     print(f"hello_throttled was called with args={args} and kwargs={kwargs}")
54     message = args[0] # as message was the first argument in the original handler
55     await message.answer("Throttled")
56
57
58 @dp.message_handler(commands=['hello'])
59 @dp.throttled(hello_throttled, rate=4)
60 async def say_hello(message: types.Message):
61     await message.answer("Hello!")
62
63
64 @dp.message_handler(commands=['help'])
65 @dp.throttled(rate=5)
66 # nothing will happen if the handler will be throttled
67 async def help_handler(message: types.Message):
68     await message.answer('Help!')
69
70
71 if __name__ == '__main__':
    start_polling(dp, skip_updates=True)

```

### 4.7.7 I18n example

Listing 7: i18n\_example.py

```

1  """
2  Internationalize your bot
3
4  Step 1: extract texts
5      # pybabel extract --input-dirs=. -o locales/mybot.pot
6
7      Some useful options:
8      - Extract texts with pluralization support
9      # -k __:1,2
10     - Add comments for translators, you can use another tag if you want (TR)
11     # --add-comments=NOTE
12     - Disable comments with string location in code
13     # --no-location
14     - Set project name
15     # --project=MySuperBot
16     - Set version
17     # --version=2.2
18
19  Step 2: create *.po files. E.g. create en, ru, uk locales.
20      # pybabel init -i locales/mybot.pot -d locales -D mybot -l en
21      # pybabel init -i locales/mybot.pot -d locales -D mybot -l ru
22      # pybabel init -i locales/mybot.pot -d locales -D mybot -l uk
23
24  Step 3: translate texts located in locales/{language}/LC_MESSAGES/mybot.po
25      To open .po file you can use basic text editor or any PO editor, e.g. https://poedit.net/
26
27  Step 4: compile translations
28      # pybabel compile -d locales -D mybot
29
30  Step 5: When you change the code of your bot you need to update po & mo files.
31      Step 5.1: regenerate pot file:
32          command from step 1
33      Step 5.2: update po files
34          # pybabel update -d locales -D mybot -i locales/mybot.pot
35      Step 5.3: update your translations
36          location and tools you know from step 3
37      Step 5.4: compile mo files
38          command from step 4
39  """
40
41  from pathlib import Path
42
43  from aiogram import Bot, Dispatcher, executor, types
44  from aiogram.contrib.middlewares.i18n import I18nMiddleware
45
46  TOKEN = 'BOT_TOKEN_HERE'
47  I18N_DOMAIN = 'mybot'
48
49  BASE_DIR = Path(__file__).parent
50  LOCALES_DIR = BASE_DIR / 'locales'
51
52  bot = Bot(TOKEN, parse_mode=types.ParseMode.HTML)

```

(continues on next page)

(continued from previous page)

```

53 dp = Dispatcher(bot)
54
55 # Setup i18n middleware
56 i18n = I18nMiddleware(I18N_DOMAIN, LOCALES_DIR)
57 dp.middleware.setup(i18n)
58
59 # Alias for gettext method
60 _ = i18n.gettext
61
62
63 @dp.message_handler(commands='start')
64 async def cmd_start(message: types.Message):
65     # Simply use `_('message')` instead of `message` and never use f-strings for
66     ↪translatable texts.
67     await message.reply(_('Hello, <b>{user}</b>!').format(user=message.from_user.full_
68     ↪name))
69
70 @dp.message_handler(commands='lang')
71 async def cmd_lang(message: types.Message, locale):
72     # For setting custom lang you have to modify i18n middleware
73     await message.reply(_('Your current language: <i>{language}</i>').
74     ↪format(language=locale))
75
76 # If you care about pluralization, here's small handler
77 # And also, there's an example of comments for translators. Most translation tools
78 ↪support them.
79
80 # Alias for gettext method, parser will understand double underscore as plural (aka
81 ↪ngettext)
82 __ = i18n.gettext
83
84
85 # some likes manager
86 LIKES_STORAGE = {'count': 0}
87
88
89 def get_likes() -> int:
90     return LIKES_STORAGE['count']
91
92
93 def increase_likes() -> int:
94     LIKES_STORAGE['count'] += 1
95     return get_likes()
96
97
98 @dp.message_handler(commands='like')
99 async def cmd_like(message: types.Message, locale):
100     likes = increase_likes()
101
102     # NOTE: This is comment for a translator
103     await message.reply(_('Aiogram has {number} like!', 'Aiogram has {number} likes!
104     ↪', likes).format(number=likes))
105
106
107 if __name__ == '__main__':
108     executor.start_polling(dp, skip_updates=True)

```

## 4.7.8 Regexp commands filter example

Listing 8: regexp\_commands\_filter\_example.py

```

1  from aiogram import Bot, types
2  from aiogram.dispatcher import Dispatcher, filters
3  from aiogram.utils import executor
4
5
6  bot = Bot(token='BOT_TOKEN_HERE', parse_mode=types.ParseMode.HTML)
7  dp = Dispatcher(bot)
8
9
10 @dp.message_handler(filters.RegexpCommandsFilter(regexp_commands=['item_([0-9]*)']))
11 async def send_welcome(message: types.Message, regexp_command):
12     await message.reply(f"You have requested an item with id <code>{regexp_command.
13     ↪group(1)}</code>")
14
15 @dp.message_handler(commands='start')
16 async def create_deeplink(message: types.Message):
17     bot_user = await bot.me
18     bot_username = bot_user.username
19     deeplink = f'https://t.me/{bot_username}?start=item_12345'
20     text = (
21         f'Either send a command /item_1234 or follow this link {deeplink} and then_
22     ↪click start\n'
23         'It also can be hidden in a inline button\n\n'
24         'Or just send <code>/start item_123</code>'
25     )
26     await message.reply(text, disable_web_page_preview=True)
27
28 if __name__ == '__main__':
29     executor.start_polling(dp, skip_updates=True)

```

## 4.7.9 Check user language

Babel is required.

Listing 9: check\_user\_language.py

```

1  """
2  Babel is required.
3  """
4
5  import logging
6
7  from aiogram import Bot, Dispatcher, executor, md, types
8
9  API_TOKEN = 'BOT TOKEN HERE'
10
11 logging.basicConfig(level=logging.INFO)
12
13
14 bot = Bot(token=API_TOKEN, parse_mode=types.ParseMode.MARKDOWN)

```

(continues on next page)



(continued from previous page)

```

15 dp = Dispatcher(bot)
16
17
18 @dp.message_handler()
19 async def check_language(message: types.Message):
20     locale = message.from_user.locale
21
22     await message.reply(md.text(
23         md.bold('Info about your language:'),
24         md.text(' ', md.bold('Code:')), md.code(locale.language)),
25         md.text(' ', md.bold('Territory:')), md.code(locale.territory or 'Unknown')),
26         md.text(' ', md.bold('Language name:')), md.code(locale.language_name)),
27         md.text(' ', md.bold('English language name:')), md.code(locale.english_name)),
28         sep='\n',
29     ))
30
31
32 if __name__ == '__main__':
33     executor.start_polling(dp, skip_updates=True)

```

## 4.7.10 Middleware and antiflood

Listing 10: middleware\_and\_antiflood.py

```

1  import asyncio
2
3  from aiogram import Bot, Dispatcher, executor, types
4  from aiogram.contrib.fsm_storage.redis import RedisStorage2
5  from aiogram.dispatcher import DEFAULT_RATE_LIMIT
6  from aiogram.dispatcher.handler import CancelHandler, current_handler
7  from aiogram.dispatcher.middlewares import BaseMiddleware
8  from aiogram.utils.exceptions import Throttled
9
10 TOKEN = 'BOT_TOKEN_HERE'
11
12 # In this example Redis storage is used
13 storage = RedisStorage2(db=5)
14
15 bot = Bot(token=TOKEN)
16 dp = Dispatcher(bot, storage=storage)
17
18
19 def rate_limit(limit: int, key=None):
20     """
21     Decorator for configuring rate limit and key in different functions.
22
23     :param limit:
24     :param key:
25     :return:
26     """
27
28     def decorator(func):
29         setattr(func, 'throttling_rate_limit', limit)
30         if key:
31             setattr(func, 'throttling_key', key)

```

(continues on next page)

(continued from previous page)

```

32     return func
33
34     return decorator
35
36
37 class ThrottlingMiddleware(BaseMiddleware):
38     """
39     Simple middleware
40     """
41
42     def __init__(self, limit=DEFAULT_RATE_LIMIT, key_prefix='antiflood_'):
43         self.rate_limit = limit
44         self.prefix = key_prefix
45         super(ThrottlingMiddleware, self).__init__()
46
47     async def on_process_message(self, message: types.Message, data: dict):
48         """
49         This handler is called when dispatcher receives a message
50
51         :param message:
52         """
53         # Get current handler
54         handler = current_handler.get()
55
56         # Get dispatcher from context
57         dispatcher = Dispatcher.get_current()
58         # If handler was configured, get rate limit and key from handler
59         if handler:
60             limit = getattr(handler, 'throttling_rate_limit', self.rate_limit)
61             key = getattr(handler, 'throttling_key', f"{self.prefix}_{handler.__name__}")
62         else:
63             limit = self.rate_limit
64             key = f"{self.prefix}_message"
65
66         # Use Dispatcher.throttle method.
67         try:
68             await dispatcher.throttle(key, rate=limit)
69         except Throttled as t:
70             # Execute action
71             await self.message_throttled(message, t)
72
73             # Cancel current handler
74             raise CancelHandler()
75
76     async def message_throttled(self, message: types.Message, throttled: Throttled):
77         """
78         Notify user only on first exceed and notify about unlocking only on last_
79         exceed
80
81         :param message:
82         :param throttled:
83         """
84         handler = current_handler.get()
85         dispatcher = Dispatcher.get_current()
86         if handler:
87             key = getattr(handler, 'throttling_key', f"{self.prefix}_{handler.__name__}")
88

```

(continues on next page)

(continued from previous page)

```

87     else:
88         key = f"{self.prefix}_message"
89
90         # Calculate how many time is left till the block ends
91         delta = throttled.rate - throttled.delta
92
93         # Prevent flooding
94         if throttled.exceeded_count <= 2:
95             await message.reply('Too many requests! ')
96
97         # Sleep.
98         await asyncio.sleep(delta)
99
100        # Check lock status
101        thr = await dispatcher.check_key(key)
102
103        # If current message is not last with current key - do not send message
104        if thr.exceeded_count == throttled.exceeded_count:
105            await message.reply('Unlocked.')
106
107
108    @dp.message_handler(commands=['start'])
109    @rate_limit(5, 'start') # this is not required but you can configure throttling_
110    ↪manager for current handler using it
111    async def cmd_test(message: types.Message):
112        # You can use this command every 5 seconds
113        await message.reply('Test passed! You can use this command every 5 seconds.')
114
115    if __name__ == '__main__':
116        # Setup middleware
117        dp.middleware.setup(ThrottlingMiddleware())
118
119        # Start long-polling
120        executor.start_polling(dp)

```

### 4.7.11 Webhook example

Listing 11: webhook\_example.py

```

1  import logging
2
3  from aiogram import Bot, types
4  from aiogram.contrib.middlewares.logging import LoggingMiddleware
5  from aiogram.dispatcher import Dispatcher
6  from aiogram.dispatcher.webhook import SendMessage
7  from aiogram.utils.executor import start_webhook
8
9
10 API_TOKEN = 'BOT_TOKEN_HERE'
11
12 # webhook settings
13 WEBHOOK_HOST = 'https://your.domain'
14 WEBHOOK_PATH = '/path/to/api'
15 WEBHOOK_URL = f"{WEBHOOK_HOST}{WEBHOOK_PATH}"

```

(continues on next page)

```

16
17 # webserver settings
18 WEBAPP_HOST = 'localhost' # or ip
19 WEBAPP_PORT = 3001
20
21 logging.basicConfig(level=logging.INFO)
22
23 bot = Bot(token=API_TOKEN)
24 dp = Dispatcher(bot)
25 dp.middleware.setup(LoggingMiddleware())
26
27
28 @dp.message_handler()
29 async def echo(message: types.Message):
30     # Regular request
31     # await bot.send_message(message.chat.id, message.text)
32
33     # or reply INTO webhook
34     return SendMessage(message.chat.id, message.text)
35
36
37 async def on_startup(dp):
38     await bot.set_webhook(WEBHOOK_URL)
39     # insert code here to run it after start
40
41
42 async def on_shutdown(dp):
43     logging.warning('Shutting down..')
44
45     # insert code here to run it before shutdown
46
47     # Remove webhook (not acceptable in some cases)
48     await bot.delete_webhook()
49
50     # Close DB connection (if used)
51     await dp.storage.close()
52     await dp.storage.wait_closed()
53
54     logging.warning('Bye!')
55
56
57 if __name__ == '__main__':
58     start_webhook(
59         dispatcher=dp,
60         webhook_path=WEBHOOK_PATH,
61         on_startup=on_startup,
62         on_shutdown=on_shutdown,
63         skip_updates=True,
64         host=WEBAPP_HOST,
65         port=WEBAPP_PORT,
66     )

```

## 4.7.12 Webhook example old

Listing 12: webhook\_example\_2.py

```

1  """
2  Example outdated
3  """
4
5  import asyncio
6  import ssl
7  import sys
8
9  from aiohttp import web
10
11 import aiogram
12 from aiogram import Bot, types
13 from aiogram.contrib.fsm_storage.memory import MemoryStorage
14 from aiogram.dispatcher import Dispatcher
15 from aiogram.dispatcher.webhook import get_new_configured_app, SendMessage
16 from aiogram.types import ChatType, ParseMode, ContentType
17 from aiogram.utils.markdown import hbold, bold, text, link
18
19 TOKEN = 'BOT TOKEN HERE'
20
21 WEBHOOK_HOST = 'example.com' # Domain name or IP address which your bot is located.
22 WEBHOOK_PORT = 443 # Telegram Bot API allows only for usage next ports: 443, 80, 88
    ↳ or 8443
23 WEBHOOK_URL_PATH = '/webhook' # Part of URL
24
25 # This options needed if you use self-signed SSL certificate
26 # Instructions: https://core.telegram.org/bots/self-signed
27 WEBHOOK_SSL_CERT = './webhook_cert.pem' # Path to the ssl certificate
28 WEBHOOK_SSL_PRIV = './webhook_pkey.pem' # Path to the ssl private key
29
30 WEBHOOK_URL = f"https://{WEBHOOK_HOST}:{WEBHOOK_PORT}{WEBHOOK_URL_PATH}"
31
32 # Web app settings:
33 # Use LAN address to listen webhooks
34 # User any available port in range from 1024 to 49151 if you're using proxy, or
    ↳ WEBHOOK_PORT if you're using direct webhook handling
35 WEBAPP_HOST = 'localhost'
36 WEBAPP_PORT = 3001
37
38 BAD_CONTENT = ContentType.PHOTO & ContentType.DOCUMENT & ContentType.STICKER &
    ↳ ContentType.AUDIO
39
40 bot = Bot(TOKEN)
41 storage = MemoryStorage()
42 dp = Dispatcher(bot, storage=storage)
43
44
45 async def cmd_start(message: types.Message):
46     # Yep. aiogram allows to respond into webhook.
47     # https://core.telegram.org/bots/api#making-requests-when-getting-updates
48     return SendMessage(chat_id=message.chat.id, text='Hi from webhook!',
49                       reply_to_message_id=message.message_id)
50

```

(continues on next page)

(continued from previous page)

```

51
52 async def cmd_about(message: types.Message):
53     # In this function markdown utils are used for formatting message text
54     return SendMessage(message.chat.id, text(
55         bold('Hi! I\'m just a simple telegram bot.'),
56         '',
57         text('I\'m powered by', bold('Python', Version(*sys.version_info[:]))),
58         text('With', link(text('aiogram', aiogram.VERSION), 'https://github.com/
↳ aiogram/aiogram')),
59         sep='\n'
60     ), parse_mode=ParseMode.MARKDOWN)
61
62
63 async def cancel(message: types.Message):
64     # Get current state context
65     state = dp.current_state(chat=message.chat.id, user=message.from_user.id)
66
67     # If current user in any state - cancel it.
68     if await state.get_state() is not None:
69         await state.set_state(state=None)
70         return SendMessage(message.chat.id, 'Current action is canceled.')
71     # Otherwise do nothing
72
73
74 async def unknown(message: types.Message):
75     """
76     Handler for unknown messages.
77     """
78     return SendMessage(message.chat.id,
79         f"I don\'t know what to do with content type `{message.content_
↳ type()}`. Sorry :c")
80
81
82 async def cmd_id(message: types.Message):
83     """
84     Return info about user.
85     """
86     if message.reply_to_message:
87         target = message.reply_to_message.from_user
88         chat = message.chat
89     elif message.forward_from and message.chat.type == ChatType.PRIVATE:
90         target = message.forward_from
91         chat = message.forward_from or message.chat
92     else:
93         target = message.from_user
94         chat = message.chat
95
96     result_msg = [hbold('Info about user:'),
97         f"First name: {target.first_name}"]
98     if target.last_name:
99         result_msg.append(f"Last name: {target.last_name}")
100     if target.username:
101         result_msg.append(f"Username: {target.mention}")
102     result_msg.append(f"User ID: {target.id}")
103
104     result_msg.extend([hbold('Chat:'),
105         f"Type: {chat.type}"],

```

(continues on next page)

(continued from previous page)

```

106         f"Chat ID: {chat.id}]]")
107     if chat.type != ChatType.PRIVATE:
108         result_msg.append(f"Title: {chat.title}")
109     else:
110         result_msg.append(f"Title: {chat.full_name}")
111     return SendMessage(message.chat.id, '\n'.join(result_msg), reply_to_message_
↳id=message.message_id,
112                         parse_mode=ParseMode.HTML)
113
114
115 async def on_startup(app):
116     # Demonstrate one of the available methods for registering handlers
117     # This command available only in main state (state=None)
118     dp.register_message_handler(cmd_start, commands=['start'])
119
120     # This handler is available in all states at any time.
121     dp.register_message_handler(cmd_about, commands=['help', 'about'], state='*')
122     dp.register_message_handler(unknown, content_types=BAD_CONTENT,
123                                func=lambda message: message.chat.type == ChatType.
↳PRIVATE)
124
125     # You are able to register one function handler for multiple conditions
126     dp.register_message_handler(cancel, commands=['cancel'], state='*')
127     dp.register_message_handler(cancel, func=lambda message: message.text.lower().
↳strip() in ['cancel'], state='*')
128
129     dp.register_message_handler(cmd_id, commands=['id'], state='*')
130     dp.register_message_handler(cmd_id, func=lambda message: message.forward_from or
131                                message.reply_to_message_
↳and
132                                message.chat.type ==
↳ChatType.PRIVATE, state='*')
133
134     # Get current webhook status
135     webhook = await bot.get_webhook_info()
136
137     # If URL is bad
138     if webhook.url != WEBHOOK_URL:
139         # If URL doesnt match current - remove webhook
140         if not webhook.url:
141             await bot.delete_webhook()
142
143         # Set new URL for webhook
144         await bot.set_webhook(WEBHOOK_URL, certificate=open(WEBHOOK_SSL_CERT, 'rb'))
145         # If you want to use free certificate signed by LetsEncrypt you need to set_
↳only URL without sending certificate.
146
147
148 async def on_shutdown(app):
149     """
150     Graceful shutdown. This method is recommended by aiohttp docs.
151     """
152     # Remove webhook.
153     await bot.delete_webhook()
154
155     # Close Redis connection.
156     await dp.storage.close()

```

(continues on next page)

(continued from previous page)

```

157     await dp.storage.wait_closed()
158
159
160 if __name__ == '__main__':
161     # Get instance of :class:`aiohttp.web.Application` with configured router.
162     app = get_new_configured_app(dispatcher=dp, path=WEBHOOK_URL_PATH)
163
164     # Setup event handlers.
165     app.on_startup.append(on_startup)
166     app.on_shutdown.append(on_shutdown)
167
168     # Generate SSL context
169     context = ssl.SSLContext(ssl.PROTOCOL_TLSv1_2)
170     context.load_cert_chain(WEBHOOK_SSL_CERT, WEBHOOK_SSL_PRIV)
171
172     # Start web-application.
173     web.run_app(app, host=WEBAPP_HOST, port=WEBAPP_PORT, ssl_context=context)
174     # Note:
175     # If you start your bot using nginx or Apache web server, SSL context is not_
176     ↪required.
177     # Otherwise you need to set `ssl_context` parameter.

```

## 4.7.13 Payments

Listing 13: payments.py

```

1  from aiogram import Bot
2  from aiogram import types
3  from aiogram.dispatcher import Dispatcher
4  from aiogram.types.message import ContentType
5  from aiogram.utils import executor
6
7
8  BOT_TOKEN = 'BOT_TOKEN_HERE'
9  PAYMENTS_PROVIDER_TOKEN = '123456789:TEST:1422'
10
11 bot = Bot(BOT_TOKEN)
12 dp = Dispatcher(bot)
13
14 # Setup prices
15 prices = [
16     types.LabeledPrice(label='Working Time Machine', amount=5750),
17     types.LabeledPrice(label='Gift wrapping', amount=500),
18 ]
19
20 # Setup shipping options
21 shipping_options = [
22     types.ShippingOption(id='instant', title='WorldWide Teleporter').add(types.
23     ↪LabeledPrice('Teleporter', 1000)),
24     types.ShippingOption(id='pickup', title='Local pickup').add(types.LabeledPrice(
25     ↪'Pickup', 300)),
26 ]
27
28 @dp.message_handler(commands=['start'])

```

(continues on next page)



(continued from previous page)

```

28 async def cmd_start(message: types.Message):
29     await bot.send_message(message.chat.id,
30                             "Hello, I'm the demo merchant bot."
31                             " I can sell you a Time Machine."
32                             " Use /buy to order one, /terms for Terms and Conditions")
33
34
35 @dp.message_handler(commands=['terms'])
36 async def cmd_terms(message: types.Message):
37     await bot.send_message(message.chat.id,
38                             'Thank you for shopping with our demo bot. We hope you_
↳like your new time machine!\n'
39                             '1. If your time machine was not delivered on time, please_
↳rethink your concept of time'
40                             ' and try again.\n'
41                             '2. If you find that your time machine is not working,_
↳kindly contact our future service'
42                             ' workshops on Trappist-1e. They will be accessible_
↳anywhere between'
43                             ' May 2075 and November 4000 C.E.\n'
44                             '3. If you would like a refund, kindly apply for one_
↳yesterday and we will have sent it'
45                             ' to you immediately.')
```

```

46
47
48 @dp.message_handler(commands=['buy'])
49 async def cmd_buy(message: types.Message):
50     await bot.send_message(message.chat.id,
51                             "Real cards won't work with me, no money will be debited_
↳from your account."
52                             " Use this test card number to pay for your Time Machine:_
↳`4242 4242 4242 4242`"
53                             "\n\nThis is your demo invoice:", parse_mode='Markdown')
54     await bot.send_invoice(message.chat.id, title='Working Time Machine',
55                             description='Want to visit your great-great-
↳grandparents?'
56                                     ' Make a fortune at the races?'
57                                     ' Shake hands with Hammurabi and take a stroll_
↳in the Hanging Gardens?'
58                                     ' Order our Working Time Machine today!',
59                             provider_token=PAYMENTS_PROVIDER_TOKEN,
60                             currency='usd',
61                             photo_url='https://telegra.ph/file/d08ff863531f10bf2ea4b.
↳jpg',
62                             photo_height=512, # !=0/None or picture won't be shown
63                             photo_width=512,
64                             photo_size=512,
65                             is_flexible=True, # True If you need to set up Shipping_
↳Fee
66                             prices=prices,
67                             start_parameter='time-machine-example',
68                             payload='HAPPY FRIDAYS COUPON')
69
70
71 @dp.shipping_query_handler(lambda query: True)
72 async def shipping(shipping_query: types.ShippingQuery):
73     await bot.answer_shipping_query(shipping_query.id, ok=True, shipping_
↳options=shipping_options,
```

(continues on next page)

(continued from previous page)

```

74         error_message='Oh, seems like our Dog couriers_
↳are having a lunch right now.'
75         ' Try again later!')
76
77
78 @dp.pre_checkout_query_handler(lambda query: True)
79 async def checkout(pre_checkout_query: types.PreCheckoutQuery):
80     await bot.answer_pre_checkout_query(pre_checkout_query.id, ok=True,
81         error_message="Aliens tried to steal your card
↳'s CVV,"
82         " but we successfully protected_
↳your credentials,"
83         " try to pay again in a few_
↳minutes, we need a small rest.")
84
85
86 @dp.message_handler(content_types=ContentTypes.SUCCESSFUL_PAYMENT)
87 async def got_payment(message: types.Message):
88     await bot.send_message(message.chat.id,
89         'Hoooooray! Thanks for payment! We will proceed your order_
↳for `{}` `{}`'
90         ' as fast as possible! Stay in touch.'
91         '\n\nUse /buy again to get a Time Machine for your friend!
↳'.format(
92             message.successful_payment.total_amount / 100, message.
↳successful_payment.currency),
93             parse_mode='Markdown')
94
95
96 if __name__ == '__main__':
97     executor.start_polling(dp, skip_updates=True)

```

#### 4.7.14 Broadcast example

Listing 14: broadcast\_example.py

```

1  import asyncio
2  import logging
3
4  from aiogram import Bot, Dispatcher, types
5  from aiogram.utils import exceptions, executor
6
7  API_TOKEN = 'BOT TOKEN HERE'
8
9  logging.basicConfig(level=logging.INFO)
10 log = logging.getLogger('broadcast')
11
12 bot = Bot(token=API_TOKEN, parse_mode=types.ParseMode.HTML)
13 dp = Dispatcher(bot)
14
15
16 def get_users():
17     """
18     Return users list
19

```

(continues on next page)

(continued from previous page)

```

20     In this example returns some random ID's
21     """
22     yield from (61043901, 78238238, 78378343, 98765431, 12345678)
23
24
25 async def send_message(user_id: int, text: str, disable_notification: bool = False) ->
26     bool:
27         """
28         Safe messages sender
29
30         :param user_id:
31         :param text:
32         :param disable_notification:
33         :return:
34         """
35         try:
36             await bot.send_message(user_id, text, disable_notification=disable_
37         notification)
38         except exceptions.BotBlocked:
39             log.error(f"Target [ID:{user_id}]: blocked by user")
40         except exceptions.ChatNotFound:
41             log.error(f"Target [ID:{user_id}]: invalid user ID")
42         except exceptions.RetryAfter as e:
43             log.error(f"Target [ID:{user_id}]: Flood limit is exceeded. Sleep {e.timeout}
44         seconds.")
45             await asyncio.sleep(e.timeout)
46             return await send_message(user_id, text) # Recursive call
47         except exceptions.UserDeactivated:
48             log.error(f"Target [ID:{user_id}]: user is deactivated")
49         except exceptions.TelegramAPIError:
50             log.exception(f"Target [ID:{user_id}]: failed")
51         else:
52             log.info(f"Target [ID:{user_id}]: success")
53             return True
54         return False
55
56
57 async def broadcaster() -> int:
58     """
59     Simple broadcaster
60
61     :return: Count of messages
62     """
63     count = 0
64     try:
65         for user_id in get_users():
66             if await send_message(user_id, '<b>Hello!</b>'):
67                 count += 1
68             await asyncio.sleep(.05) # 20 messages per second (Limit: 30 messages
69         per second)
70     finally:
71         log.info(f"{count} messages successful sent.")
72
73     return count
74
75
76 if __name__ == '__main__':

```

(continues on next page)

(continued from previous page)

```

73     # Execute broadcaster
74     executor.start(dp, broadcaster())

```

## 4.7.15 Media group

Listing 15: media\_group.py

```

1  import asyncio
2
3  from aiogram import Bot, Dispatcher, executor, filters, types
4
5
6  API_TOKEN = 'BOT_TOKEN_HERE'
7
8  bot = Bot(token=API_TOKEN)
9  dp = Dispatcher(bot)
10
11
12 @dp.message_handler(filters.CommandStart())
13 async def send_welcome(message: types.Message):
14     # So... At first I want to send something like this:
15     await message.reply("Do you want to see many pussies? Are you ready?")
16
17     # Wait a little...
18     await asyncio.sleep(1)
19
20     # Good bots should send chat actions...
21     await types.ChatActions.upload_photo()
22
23     # Create media group
24     media = types.MediaGroup()
25
26     # Attach local file
27     media.attach_photo(types.InputFile('data/cat.jpg'), 'Cat!')
28     # More local files and more cats!
29     media.attach_photo(types.InputFile('data/cats.jpg'), 'More cats!')
30
31     # You can also use URL's
32     # For example: get random puss:
33     media.attach_photo('http://lorempixel.com/400/200/cats/', 'Random cat.')
34
35     # And you can also use file ID:
36     # media.attach_photo('<file_id>', 'cat-cat-cat.')
37
38     # Done! Send media group
39     await message.reply_media_group(media=media)
40
41
42 if __name__ == '__main__':
43     executor.start_polling(dp, skip_updates=True)

```

### 4.7.16 Local server

Listing 16: local\_server.py

```
1 import logging
2
3 from aiogram import Bot, Dispatcher, executor, types
4 from aiogram.bot.api import TelegramAPIServer
5 from aiogram.types import ContentType
6
7 API_TOKEN = 'BOT TOKEN HERE'
8
9 # Configure logging
10 logging.basicConfig(level=logging.INFO)
11
12 # Create private Bot API server endpoints wrapper
13 local_server = TelegramAPIServer.from_base('http://localhost')
14
15 # Initialize bot with using local server
16 bot = Bot(token=API_TOKEN, server=local_server)
17 # ... and dispatcher
18 dp = Dispatcher(bot)
19
20
21 @dp.message_handler(content_types=ContentType.ANY)
22 async def echo(message: types.Message):
23     await message.copy_to(message.chat.id)
24
25
26 if __name__ == '__main__':
27     executor.start_polling(dp, skip_updates=True)
```

## 4.8 Contribution

TODO

## 4.9 Links

TODO



## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`





## PYTHON MODULE INDEX

### a

- `aiogram.bot.api`, [57](#)
- `aiogram.utils.auth_widget`, [185](#)
- `aiogram.utils.deep_linking`, [199](#)
- `aiogram.utils.deprecated`, [197](#)
- `aiogram.utils.emoji`, [199](#)
- `aiogram.utils.exceptions`, [188](#)
- `aiogram.utils.executor`, [186](#)
- `aiogram.utils.helper`, [196](#)
- `aiogram.utils.json`, [199](#)
- `aiogram.utils.markdown`, [194](#)
- `aiogram.utils.parts`, [198](#)
- `aiogram.utils.payload`, [198](#)



## A

AbstractFilter (class in aiogram.dispatcher.filters), 168  
 add() (aiogram.types.inline\_keyboard.InlineKeyboardMarkup method), 68  
 add() (aiogram.types.reply\_keyboard.ReplyKeyboardMarkup method), 72  
 add() (aiogram.types.shipping\_option.ShippingOption method), 124  
 add\_sticker\_to\_set() (aiogram.bot.bot.Bot method), 51  
 AdminFilter (class in aiogram.dispatcher.filters), 167  
 aiogram.bot.api module, 57  
 aiogram.utils.auth\_widget module, 185  
 aiogram.utils.deep\_linking module, 199  
 aiogram.utils.deprecated module, 197  
 aiogram.utils.emoji module, 199  
 aiogram.utils.exceptions module, 188  
 aiogram.utils.executor module, 186  
 aiogram.utils.helper module, 196  
 aiogram.utils.json module, 199  
 aiogram.utils.markdown module, 194  
 aiogram.utils.parts module, 198  
 aiogram.utils.payload module, 198  
 AIOGramWarning, 191  
 AllowedUpdates (class in aiogram.types.update), 121  
 Animation (class in aiogram.types.animation), 89  
 answer() (aiogram.types.callback\_query.CallbackQuery method), 65  
 answer() (aiogram.types.inline\_query.InlineQuery method), 88  
 answer() (aiogram.types.message.Message method), 126  
 answer\_animation() (aiogram.types.message.Message method), 129  
 answer\_audio() (aiogram.types.message.Message method), 128  
 answer\_callback\_query() (aiogram.bot.bot.Bot method), 45  
 answer\_contact() (aiogram.types.message.Message method), 136  
 answer\_dice() (aiogram.types.message.Message method), 138  
 answer\_document() (aiogram.types.message.Message method), 130  
 answer\_inline\_query() (aiogram.bot.bot.Bot method), 52  
 answer\_location() (aiogram.types.message.Message method), 134  
 answer\_media\_group() (aiogram.types.message.Message method), 134  
 answer\_photo() (aiogram.types.message.Message method), 127  
 answer\_poll() (aiogram.types.message.Message method), 137  
 answer\_pre\_checkout\_query() (aiogram.bot.bot.Bot method), 55  
 answer\_shipping\_query() (aiogram.bot.bot.Bot method), 54  
 answer\_sticker() (aiogram.types.message.Message method), 136  
 answer\_venue() (aiogram.types.message.Message method), 135  
 answer\_video() (aiogram.types.message.Message method), 131  
 answer\_video\_note() (aiogram.types.message.Message method),

- 133
- `answer_voice()` (*aiogram.types.message.Message* method), 132
- `api_url()` (*aiogram.bot.api.TelegramAPIServer* method), 57
- `args` (*aiogram.dispatcher.filters.Command.CommandObj* attribute), 160
- `as_json()` (*aiogram.types.base.TelegramObject* method), 59
- `async_task()` (*aiogram.Dispatcher* method), 183
- `attach()` (*aiogram.types.input\_media.MediaGroup* method), 92
- `attach_many()` (*aiogram.types.input\_media.MediaGroup* method), 92
- `attach_photo()` (*aiogram.types.input\_media.MediaGroup* method), 92
- `attach_video()` (*aiogram.types.input\_media.MediaGroup* method), 92
- `Audio` (class in *aiogram.types.audio*), 82
- `AuthWidgetData` (class in *aiogram.types.auth\_widget\_data*), 158
- ## B
- `BadRequest`, 191
- `BadWebhook`, 193
- `BadWebhookAddrInfo`, 193
- `BadWebhookNoAddressAssociatedWithHostname`, 193
- `BadWebhookPort`, 193
- `BaseBot` (class in *aiogram.bot.base*), 17
- `BaseField` (class in *aiogram.types.fields*), 60
- `bind()` (*aiogram.dispatcher.filters.FiltersFactory* method), 158
- `bind_filter()` (*aiogram.Dispatcher* method), 184
- `bold()` (in module *aiogram.utils.markdown*), 194
- `Bot` (class in *aiogram.bot.bot*), 19
- `BotBlocked`, 193
- `BotDomainInvalid`, 193
- `BotKicked`, 193
- `BoundFilter` (class in *aiogram.dispatcher.filters*), 169
- `ButtonDataInvalid`, 192
- `ButtonURLInvalid`, 192
- ## C
- `calc_timeout()` (*aiogram.types.chat.ChatActions* class method), 80
- `callback_query_handler()` (*aiogram.Dispatcher* method), 180
- `CallbackGame` (class in *aiogram.types.callback\_game*), 71
- `CallbackQuery` (class in *aiogram.types.callback\_query*), 65
- `CantDemoteChatCreator`, 192
- `CantGetUpdates`, 193
- `CantInitiateConversation`, 193
- `CantParseEntities`, 193
- `CantParseUrl`, 193
- `CantRestrictSelf`, 192
- `CantTalkWithBots`, 193
- `channel_post_handler()` (*aiogram.Dispatcher* method), 177
- `Chat` (class in *aiogram.types.chat*), 73
- `ChatActions` (class in *aiogram.types.chat*), 80
- `ChatAdminRequired`, 192
- `ChatDescriptionIsNotModified`, 192
- `ChatIdIsEmpty`, 192
- `ChatMember` (class in *aiogram.types.chat\_member*), 123
- `ChatMemberStatus` (class in *aiogram.types.chat\_member*), 124
- `ChatNotFound`, 192
- `ChatPhoto` (class in *aiogram.types.chat\_photo*), 124
- `ChatType` (class in *aiogram.types.chat*), 79
- `ChatTypeFilter` (class in *aiogram.dispatcher.filters*), 168
- `check()` (*aiogram.dispatcher.filters.AbstractFilter* method), 168
- `check()` (*aiogram.dispatcher.filters.AdminFilter* method), 167
- `check()` (*aiogram.dispatcher.filters.builtin.IDFilter* method), 166
- `check()` (*aiogram.dispatcher.filters.ChatTypeFilter* method), 168
- `check()` (*aiogram.dispatcher.filters.Command* method), 160
- `check()` (*aiogram.dispatcher.filters.CommandStart* method), 161
- `check()` (*aiogram.dispatcher.filters.ContentTypeFilter* method), 165
- `check()` (*aiogram.dispatcher.filters.ExceptionsFilter* method), 166
- `check()` (*aiogram.dispatcher.filters.ForwardedMessageFilter* method), 167
- `check()` (*aiogram.dispatcher.filters.HashTag* method), 164
- `check()` (*aiogram.dispatcher.filters.IsReplyFilter* method), 167
- `check()` (*aiogram.dispatcher.filters.IsSenderContact* method), 165
- `check()` (*aiogram.dispatcher.filters.Regexp* method), 164
- `check()` (*aiogram.dispatcher.filters.RegexpCommandsFilter* method), 165
- `check()` (*aiogram.dispatcher.filters.StateFilter* method), 166
- `check()` (*aiogram.dispatcher.filters.Text* method), 164
- `check_integrity()` (in module *aiogram.utils.auth\_widget*), 185

- [check\\_key\(\)](#) (*aiogram.Dispatcher* method), 183  
[check\\_result\(\)](#) (in module *aiogram.bot.api*), 58  
[check\\_signature\(\)](#) (in module *aiogram.utils.auth\_widget*), 185  
[check\\_token\(\)](#) (in module *aiogram.bot.api*), 58  
[check\\_token\(\)](#) (in module *aiogram.utils.auth\_widget*), 185  
[chosen\\_inline\\_handler\(\)](#) (*aiogram.Dispatcher* method), 179  
[ChosenInlineResult](#) (class in *aiogram.types.chosen\_inline\_result*), 122  
[clean\(\)](#) (*aiogram.types.base.TelegramObject* method), 59  
[close\(\)](#) (*aiogram.bot.base.BaseBot* method), 18  
[close\\_bot\(\)](#) (*aiogram.bot.bot.Bot* method), 21  
[code\(\)](#) (in module *aiogram.utils.markdown*), 195  
[command](#) (*aiogram.dispatcher.filters.Command.CommandObj* attribute), 160  
[Command](#) (class in *aiogram.dispatcher.filters*), 159  
[Command.CommandObj](#) (class in *aiogram.dispatcher.filters*), 160  
[CommandHelp](#) (class in *aiogram.dispatcher.filters*), 161  
[CommandPrivacy](#) (class in *aiogram.dispatcher.filters*), 162  
[CommandSettings](#) (class in *aiogram.dispatcher.filters*), 162  
[CommandStart](#) (class in *aiogram.dispatcher.filters*), 161  
[compose\\_data\(\)](#) (in module *aiogram.bot.api*), 58  
[ConflictError](#), 193  
[Contact](#) (class in *aiogram.types.contact*), 125  
[ContentType](#) (class in *aiogram.types.message*), 155  
[ContentTypeFilter](#) (class in *aiogram.dispatcher.filters*), 165  
[ContentTypes](#) (class in *aiogram.types.message*), 156  
[copy\\_message\(\)](#) (*aiogram.bot.bot.Bot* method), 23  
[create\(\)](#) (*aiogram.types.force\_reply.ForceReply* class method), 82  
[create\\_new\\_sticker\\_set\(\)](#) (*aiogram.bot.bot.Bot* method), 50  
[CurrencyTotalAmountInvalid](#), 193  
[current\\_state\(\)](#) (*aiogram.Dispatcher* method), 183
- ## D
- [DateTimeField](#) (class in *aiogram.types.fields*), 62  
[decode\\_payload\(\)](#) (in module *aiogram.utils.deep\_linking*), 199  
[default](#) (*aiogram.dispatcher.filters.BoundFilter* attribute), 169  
[delete\(\)](#) (*aiogram.types.message.Message* method), 154  
[delete\\_chat\\_photo\(\)](#) (*aiogram.bot.bot.Bot* method), 42  
[delete\\_chat\\_sticker\\_set\(\)](#) (*aiogram.bot.bot.Bot* method), 45  
[delete\\_from\\_set\(\)](#) (*aiogram.types.sticker.Sticker* method), 88  
[delete\\_message\(\)](#) (*aiogram.bot.bot.Bot* method), 49  
[delete\\_photo\(\)](#) (*aiogram.types.chat.Chat* method), 74  
[delete\\_reply\\_markup\(\)](#) (*aiogram.types.message.Message* method), 153  
[delete\\_sticker\\_from\\_set\(\)](#) (*aiogram.bot.bot.Bot* method), 52  
[delete\\_sticker\\_set\(\)](#) (*aiogram.types.chat.Chat* method), 78  
[delete\\_webhook\(\)](#) (*aiogram.bot.bot.Bot* method), 21  
[deprecated\(\)](#) (in module *aiogram.utils.deprecated*), 197  
[DeprecatedReadOnlyClassVar](#) (class in *aiogram.utils.deprecated*), 197  
[deserialize\(\)](#) (*aiogram.types.fields.BaseField* method), 60  
[deserialize\(\)](#) (*aiogram.types.fields.DateTimeField* method), 63  
[deserialize\(\)](#) (*aiogram.types.fields.Field* method), 61  
[deserialize\(\)](#) (*aiogram.types.fields.ListField* method), 62  
[deserialize\(\)](#) (*aiogram.types.fields.ListOfLists* method), 62  
[deserialize\(\)](#) (*aiogram.types.fields.TextField* method), 63  
[Dispatcher](#) (class in *aiogram*), 173  
[do\(\)](#) (*aiogram.types.chat.Chat* method), 79  
[Document](#) (class in *aiogram.types.document*), 82  
[download\(\)](#) (*aiogram.types.mixins.Downloadable* method), 63  
[download\\_big\(\)](#) (*aiogram.types.chat\_photo.ChatPhoto* method), 125  
[download\\_file\(\)](#) (*aiogram.bot.base.BaseBot* method), 18  
[download\\_file\\_by\\_id\(\)](#) (*aiogram.bot.bot.Bot* method), 19  
[download\\_small\(\)](#) (*aiogram.types.chat\_photo.ChatPhoto* method), 124  
[Downloadable](#) (class in *aiogram.types.mixins*), 63
- ## E
- [edit\\_caption\(\)](#) (*aiogram.types.message.Message* method), 152  
[edit\\_live\\_location\(\)](#) (*aiogram.types.message.Message* method), 153

`edit_media()` (*aiogram.types.message.Message* method), 152  
`edit_message_caption()` (*aiogram.bot.bot.Bot* method), 47  
`edit_message_live_location()` (*aiogram.bot.bot.Bot* method), 32  
`edit_message_media()` (*aiogram.bot.bot.Bot* method), 47  
`edit_message_reply_markup()` (*aiogram.bot.bot.Bot* method), 48  
`edit_message_text()` (*aiogram.bot.bot.Bot* method), 46  
`edit_reply_markup()` (*aiogram.types.message.Message* method), 153  
`edit_text()` (*aiogram.types.message.Message* method), 152  
`edited_channel_post_handler()` (*aiogram.Dispatcher* method), 178  
`edited_message_handler()` (*aiogram.Dispatcher* method), 176  
`encode_payload()` (in module *aiogram.utils.deep\_linking*), 199  
`EncryptedCredentials` (class in *aiogram.types.encrypted\_credentials*), 64  
`EncryptedPassportElement` (class in *aiogram.types.encrypted\_passport\_element*), 70  
`errors_handler()` (*aiogram.Dispatcher* method), 182  
`escape_md()` (in module *aiogram.utils.markdown*), 194  
`ExceptionsFilter` (class in *aiogram.dispatcher.filters*), 166  
`Executor` (class in *aiogram.utils.executor*), 187  
`export()` (*aiogram.types.fields.BaseField* method), 61  
`export_chat_invite_link()` (*aiogram.bot.bot.Bot* method), 41  
`export_invite_link()` (*aiogram.types.chat.Chat* method), 79

## F

`Field` (class in *aiogram.types.fields*), 61  
`File` (class in *aiogram.types.file*), 71  
`file_url()` (*aiogram.bot.api.TelegramAPIServer* method), 58  
`FileIsTooBig`, 192  
`Filter` (class in *aiogram.dispatcher.filters*), 169  
`filter_payload()` (in module *aiogram.utils.deep\_linking*), 199  
`FiltersFactory` (class in *aiogram.dispatcher.filters*), 158  
`find_location()` (*aiogram.types.chat.ChatActions* class method), 81

`ForceReply` (class in *aiogram.types.force\_reply*), 82  
`forward()` (*aiogram.types.message.Message* method), 151  
`forward_message()` (*aiogram.bot.bot.Bot* method), 22  
`ForwardedMessageFilter` (class in *aiogram.dispatcher.filters*), 167  
`from_url()` (*aiogram.types.input\_file.InputFile* class method), 115  
`FSMStorageWarning`, 191  
`full_name()` (*aiogram.types.user.User* property), 69

## G

`Game` (class in *aiogram.types.game*), 70  
`GameHighScore` (class in *aiogram.types.game\_high\_score*), 87  
`generate_hash()` (in module *aiogram.utils.auth\_widget*), 185  
`generate_payload()` (in module *aiogram.utils.payload*), 198  
`get_administrators()` (*aiogram.types.chat.Chat* method), 78  
`get_args()` (*aiogram.types.message.Message* method), 126  
`get_chat()` (*aiogram.bot.bot.Bot* method), 44  
`get_chat_administrators()` (*aiogram.bot.bot.Bot* method), 44  
`get_chat_member()` (*aiogram.bot.bot.Bot* method), 44  
`get_chat_members_count()` (*aiogram.bot.bot.Bot* method), 44  
`get_command()` (*aiogram.types.message.Message* method), 126  
`get_file()` (*aiogram.bot.bot.Bot* method), 38  
`get_file()` (*aiogram.types.input\_file.InputFile* method), 115  
`get_file()` (*aiogram.types.mixins.Downloadable* method), 63  
`get_filename()` (*aiogram.types.input\_file.InputFile* method), 115  
`get_full_command()` (*aiogram.types.message.Message* method), 126  
`get_game_high_scores()` (*aiogram.bot.bot.Bot* method), 57  
`get_me()` (*aiogram.bot.bot.Bot* method), 21  
`get_member()` (*aiogram.types.chat.Chat* method), 78  
`get_members_count()` (*aiogram.types.chat.Chat* method), 78  
`get_my_commands()` (*aiogram.bot.bot.Bot* method), 46  
`get_start_link()` (in module *aiogram.utils.deep\_linking*), 199

[get\\_startgroup\\_link\(\)](#) (in module [aiogram.utils.deep\\_linking](#)), 199  
[get\\_sticker\\_set\(\)](#) ([aiogram.bot.bot.Bot](#) method), 50  
[get\\_text\(\)](#) ([aiogram.types.message\\_entity.MessageEntity](#) method), 66  
[get\\_updates\(\)](#) ([aiogram.bot.bot.Bot](#) method), 20  
[get\\_url\(\)](#) ([aiogram.types.chat.Chat](#) method), 73  
[get\\_url\(\)](#) ([aiogram.types.mixins.Downloadable](#) method), 64  
[get\\_user\\_profile\\_photos\(\)](#) ([aiogram.bot.bot.Bot](#) method), 38  
[get\\_value\(\)](#) ([aiogram.types.fields.BaseField](#) method), 60  
[get\\_webhook\\_info\(\)](#) ([aiogram.bot.bot.Bot](#) method), 21  
[GroupDeactivated](#), 192  
[guess\\_filename\(\)](#) (in module [aiogram.bot.api](#)), 58

## H

[HashTag](#) (class in [aiogram.dispatcher.filters](#)), 164  
[hbold\(\)](#) (in module [aiogram.utils.markdown](#)), 194  
[hcode\(\)](#) (in module [aiogram.utils.markdown](#)), 195  
[hide\\_link\(\)](#) (in module [aiogram.utils.markdown](#)), 196  
[hitalic\(\)](#) (in module [aiogram.utils.markdown](#)), 194  
[hlink\(\)](#) (in module [aiogram.utils.markdown](#)), 196  
[hpre\(\)](#) (in module [aiogram.utils.markdown](#)), 195  
[hstrikethrough\(\)](#) (in module [aiogram.utils.markdown](#)), 196  
[html\\_text\(\)](#) ([aiogram.types.message.Message](#) property), 126  
[hunderline\(\)](#) (in module [aiogram.utils.markdown](#)), 195

## I

[IDFilter](#) (class in [aiogram.dispatcher.filters.builtin](#)), 166  
[inline\\_handler\(\)](#) ([aiogram.Dispatcher](#) method), 178  
[InlineKeyboardButton](#) (class in [aiogram.types.inline\\_keyboard](#)), 69  
[InlineKeyboardExpected](#), 191  
[InlineKeyboardMarkup](#) (class in [aiogram.types.inline\\_keyboard](#)), 68  
[InlineQuery](#) (class in [aiogram.types.inline\\_query](#)), 88  
[InlineQueryResult](#) (class in [aiogram.types.inline\\_query\\_result](#)), 93  
[InlineQueryResultArticle](#) (class in [aiogram.types.inline\\_query\\_result](#)), 94  
[InlineQueryResultAudio](#) (class in [aiogram.types.inline\\_query\\_result](#)), 99  
[InlineQueryResultCachedAudio](#) (class in [aiogram.types.inline\\_query\\_result](#)), 114  
[InlineQueryResultCachedDocument](#) (class in [aiogram.types.inline\\_query\\_result](#)), 111  
[InlineQueryResultCachedGif](#) (class in [aiogram.types.inline\\_query\\_result](#)), 108  
[InlineQueryResultCachedMpeg4Gif](#) (class in [aiogram.types.inline\\_query\\_result](#)), 109  
[InlineQueryResultCachedPhoto](#) (class in [aiogram.types.inline\\_query\\_result](#)), 107  
[InlineQueryResultCachedSticker](#) (class in [aiogram.types.inline\\_query\\_result](#)), 110  
[InlineQueryResultCachedVideo](#) (class in [aiogram.types.inline\\_query\\_result](#)), 112  
[InlineQueryResultCachedVoice](#) (class in [aiogram.types.inline\\_query\\_result](#)), 113  
[InlineQueryResultContact](#) (class in [aiogram.types.inline\\_query\\_result](#)), 105  
[InlineQueryResultDocument](#) (class in [aiogram.types.inline\\_query\\_result](#)), 101  
[InlineQueryResultGame](#) (class in [aiogram.types.inline\\_query\\_result](#)), 106  
[InlineQueryResultGif](#) (class in [aiogram.types.inline\\_query\\_result](#)), 96  
[InlineQueryResultLocation](#) (class in [aiogram.types.inline\\_query\\_result](#)), 103  
[InlineQueryResultMpeg4Gif](#) (class in [aiogram.types.inline\\_query\\_result](#)), 97  
[InlineQueryResultPhoto](#) (class in [aiogram.types.inline\\_query\\_result](#)), 95  
[InlineQueryResultVenue](#) (class in [aiogram.types.inline\\_query\\_result](#)), 104  
[InlineQueryResultVideo](#) (class in [aiogram.types.inline\\_query\\_result](#)), 98  
[InlineQueryResultVoice](#) (class in [aiogram.types.inline\\_query\\_result](#)), 100  
[InputContactMessageContent](#) (class in [aiogram.types.input\\_message\\_content](#)), 117  
[InputFile](#) (class in [aiogram.types.input\\_file](#)), 115  
[InputLocationMessageContent](#) (class in [aiogram.types.input\\_message\\_content](#)), 118  
[InputMedia](#) (class in [aiogram.types.input\\_media](#)), 89  
[InputMediaAnimation](#) (class in [aiogram.types.input\\_media](#)), 90  
[InputMediaAudio](#) (class in [aiogram.types.input\\_media](#)), 91  
[InputMediaDocument](#) (class in [aiogram.types.input\\_media](#)), 90  
[InputMediaPhoto](#) (class in [aiogram.types.input\\_media](#)), 91  
[InputMediaVideo](#) (class in [aiogram.types.input\\_media](#)), 92  
[InputMessageContent](#) (class in [aiogram.types.input\\_message\\_content](#)), 117



- InputTextMessageContent (class in *aiogram.types.input\_message\_content*), 119  
 InputVenueMessageContent (class in *aiogram.types.input\_message\_content*), 120  
 insert() (*aiogram.types.inline\_keyboard.InlineKeyboardMarkup* method), 68  
 insert() (*aiogram.types.reply\_keyboard.ReplyKeyboardMarkup* method), 72  
 InvalidHttpUrlContent, 192  
 InvalidPeerID, 192  
 InvalidQueryID, 192  
 InvalidStickersSet, 192  
 InvalidUserId, 192  
 Invoice (class in *aiogram.types.invoice*), 157  
 is\_channel() (*aiogram.types.chat.ChatType* class method), 80  
 is\_command() (*aiogram.types.message.Message* method), 125  
 is\_forward() (*aiogram.types.message.Message* method), 125  
 is\_group() (*aiogram.types.chat.ChatType* class method), 79  
 is\_group\_or\_super\_group() (*aiogram.types.chat.ChatType* class method), 80  
 is\_polling() (*aiogram.Dispatcher* method), 174  
 is\_private() (*aiogram.types.chat.ChatType* class method), 79  
 is\_super\_group() (*aiogram.types.chat.ChatType* class method), 80  
 IsReplyFilter (class in *aiogram.dispatcher.filters*), 167  
 IsSenderContact (class in *aiogram.dispatcher.filters*), 165  
 italic() (in module *aiogram.utils.markdown*), 194  
 Item (class in *aiogram.utils.helper*), 196  
 ItemsList (class in *aiogram.utils.helper*), 196  
 iter\_keys() (*aiogram.types.base.TelegramObject* method), 60  
 iter\_values() (*aiogram.types.base.TelegramObject* method), 60
- ## K
- key (*aiogram.dispatcher.filters.BoundFilter* attribute), 169  
 KeyboardButton (class in *aiogram.types.reply\_keyboard*), 72  
 kick() (*aiogram.types.chat.Chat* method), 74  
 kick\_chat\_member() (*aiogram.bot.bot.Bot* method), 38
- ## L
- LabeledPrice (class in *aiogram.types.labeled\_price*), 71  
 leave() (*aiogram.types.chat.Chat* method), 78  
 leave\_chat() (*aiogram.bot.bot.Bot* method), 43  
 link() (*aiogram.types.message.Message* method), 126  
 link() (in module *aiogram.utils.markdown*), 196  
 Markup (class in *aiogram.types.fields*), 61  
 ListItem (class in *aiogram.utils.helper*), 196  
 Markup (class in *aiogram.types.fields*), 62  
 locale() (*aiogram.types.user.User* property), 69  
 Location (class in *aiogram.types.location*), 89  
 log\_out() (*aiogram.bot.bot.Bot* method), 21
- ## M
- MaskPosition (class in *aiogram.types.mask\_position*), 157  
 md\_text() (*aiogram.types.message.Message* property), 126  
 me() (*aiogram.bot.bot.Bot* property), 19  
 MediaGroup (class in *aiogram.types.input\_media*), 92  
 MemoryStorage (class in *aiogram.contrib.fsm\_storage.memory*), 170  
 mention (*aiogram.dispatcher.filters.Command.CommandObj* attribute), 160  
 mention() (*aiogram.types.chat.Chat* property), 73  
 mention() (*aiogram.types.user.User* property), 69  
 mentioned() (*aiogram.dispatcher.filters.Command.CommandObj* property), 160  
 Message (class in *aiogram.types.message*), 125  
 message\_handler() (*aiogram.Dispatcher* method), 175  
 MessageCantBeDeleted, 191  
 MessageCantBeEdited, 191  
 MessageCantBeForwarded, 191  
 MessageEntity (class in *aiogram.types.message\_entity*), 66  
 MessageEntityType (class in *aiogram.types.message\_entity*), 67  
 MessageError, 191  
 MessageIdentifierNotSpecified, 191  
 MessageIsNotAPoll, 192  
 MessageIsTooLong, 191  
 MessageNotModified, 191  
 MessageTextIsEmpty, 191  
 MessageToDeleteNotFound, 191  
 MessageToEditNotFound, 191  
 MessageToForwardNotFound, 191  
 MessageToPinNotFound, 191  
 MessageToReplyNotFound, 191  
 MessageWithPollNotFound, 192  
 MetaTelegramObject (class in *aiogram.types.base*), 59  
 MethodIsNotAvailable, 193  
 MethodNotAvailableInPrivateChats, 192  
 MethodNotKnown, 193  
 Methods (class in *aiogram.bot.api*), 58



MigrateToChat, 193

module

- aiogram.bot.api, 57
- aiogram.utils.auth\_widget, 185
- aiogram.utils.deep\_linking, 199
- aiogram.utils.deprecated, 197
- aiogram.utils.emoji, 199
- aiogram.utils.exceptions, 188
- aiogram.utils.executor, 186
- aiogram.utils.helper, 196
- aiogram.utils.json, 199
- aiogram.utils.markdown, 194
- aiogram.utils.parts, 198
- aiogram.utils.payload, 198

MongoStorage (class in *aiogram.contrib.fsm\_storage.mongo*), 170

## N

NeedAdministratorRightsInTheChannel, 192

NetworkError, 193

NoStickerInRequest, 192

NotEnoughRightsToPinMessage, 192

NotEnoughRightsToRestrict, 193

NotFound, 193

## O

ObjectExpectedAsReplyMarkup, 191

on\_shutdown() (aiogram.utils.executor.Executor method), 187

on\_startup() (aiogram.utils.executor.Executor method), 187

OrderInfo (class in *aiogram.types.order\_info*), 87

## P

paginate() (in module *aiogram.utils.parts*), 198

parse() (aiogram.types.auth\_widget\_data.AuthWidgetData class method), 158

parse() (aiogram.types.message\_entity.MessageEntity method), 66

parse\_entities() (aiogram.types.message.Message method), 126

ParseMode (class in *aiogram.types.message*), 156

PassportData (class in *aiogram.types.passport\_data*), 68

PassportElementError (class in *aiogram.types.passport\_element\_error*), 83

PassportElementErrorDataField (class in *aiogram.types.passport\_element\_error*), 83

PassportElementErrorFile (class in *aiogram.types.passport\_element\_error*), 84

PassportElementErrorFiles (class in *aiogram.types.passport\_element\_error*), 84

PassportElementErrorFrontSide (class in *aiogram.types.passport\_element\_error*), 85

PassportElementErrorReverseSide (class in *aiogram.types.passport\_element\_error*), 85

PassportElementErrorSelfie (class in *aiogram.types.passport\_element\_error*), 86

PassportFile (class in *aiogram.types.passport\_file*), 123

PaymentProviderInvalid, 193

PhotoAsInputFileRequired, 192

PhotoDimensions, 193

PhotoSize (class in *aiogram.types.photo\_size*), 121

pin() (aiogram.types.message.Message method), 154

pin\_chat\_message() (aiogram.bot.bot.Bot method), 43

pin\_message() (aiogram.types.chat.Chat method), 77

poll\_answer\_handler() (aiogram.Dispatcher method), 182

poll\_handler() (aiogram.Dispatcher method), 181

PollCanBeRequestedInPrivateChatsOnly, 192

PollCantBeStopped, 191

PollCantHaveMoreOptions, 192

PollError, 191

PollHasAlreadyBeenClosed, 191

PollMustHaveMoreOptions, 192

PollOptionsLengthTooLong, 192

PollOptionsMustBeNonEmpty, 192

PollQuestionLengthTooLong, 192

PollQuestionMustBeNonEmpty, 192

PollsCantBeSentToPrivateChats, 192

PollSizeError, 192

pre() (in module *aiogram.utils.markdown*), 195

pre\_checkout\_query\_handler() (aiogram.Dispatcher method), 181

PreCheckoutQuery (class in *aiogram.types.pre\_checkout\_query*), 116

prefix (aiogram.dispatcher.filters.Command.CommandObj attribute), 160

prepare\_arg() (in module *aiogram.utils.payload*), 198

process\_update() (aiogram.Dispatcher method), 173

process\_updates() (aiogram.Dispatcher method), 173

promote() (aiogram.types.chat.Chat method), 76

promote\_chat\_member() (aiogram.bot.bot.Bot method), 40

props() (aiogram.types.base.TelegramObject prop-

erty), 59  
 props\_aliases() (aiogram.types.base.TelegramObject  
 property), 59

## Q

quote\_html() (in module aiogram.utils.markdown),  
 194

## R

record\_audio() (aiogram.types.chat.ChatActions  
 class method), 81  
 record\_video() (aiogram.types.chat.ChatActions  
 class method), 81  
 record\_video\_note()  
 (aiogram.types.chat.ChatActions class  
 method), 81  
 RedisStorage (class in  
 aiogram.contrib.fsm\_storage.redis), 170  
 Regex (class in aiogram.dispatcher.filters), 164  
 RegexCommandsFilter (class in  
 aiogram.dispatcher.filters), 165  
 register\_callback\_query\_handler()  
 (aiogram.Dispatcher method), 179  
 register\_channel\_post\_handler()  
 (aiogram.Dispatcher method), 177  
 register\_chosen\_inline\_handler()  
 (aiogram.Dispatcher method), 179  
 register\_edited\_channel\_post\_handler()  
 (aiogram.Dispatcher method), 177  
 register\_edited\_message\_handler()  
 (aiogram.Dispatcher method), 176  
 register\_errors\_handler()  
 (aiogram.Dispatcher method), 182  
 register\_inline\_handler()  
 (aiogram.Dispatcher method), 178  
 register\_message\_handler()  
 (aiogram.Dispatcher method), 174  
 register\_poll\_answer\_handler()  
 (aiogram.Dispatcher method), 182  
 register\_poll\_handler() (aiogram.Dispatcher  
 method), 181  
 register\_pre\_checkout\_query\_handler()  
 (aiogram.Dispatcher method), 180  
 register\_shipping\_query\_handler()  
 (aiogram.Dispatcher method), 180  
 release\_key() (aiogram.Dispatcher method), 183  
 renamed\_argument() (in module  
 aiogram.utils.deprecated), 197  
 reply() (aiogram.types.message.Message method),  
 139  
 reply\_animation()  
 (aiogram.types.message.Message method),  
 141

reply\_audio() (aiogram.types.message.Message  
 method), 140  
 reply\_contact() (aiogram.types.message.Message  
 method), 148  
 reply\_dice() (aiogram.types.message.Message  
 method), 151  
 reply\_document() (aiogram.types.message.Message  
 method), 143  
 reply\_location() (aiogram.types.message.Message  
 method), 147  
 reply\_media\_group()  
 (aiogram.types.message.Message method),  
 146  
 reply\_photo() (aiogram.types.message.Message  
 method), 139  
 reply\_poll() (aiogram.types.message.Message  
 method), 149  
 reply\_sticker() (aiogram.types.message.Message  
 method), 150  
 reply\_venue() (aiogram.types.message.Message  
 method), 147  
 reply\_video() (aiogram.types.message.Message  
 method), 144  
 reply\_video\_note()  
 (aiogram.types.message.Message method),  
 146  
 reply\_voice() (aiogram.types.message.Message  
 method), 145  
 ReplyKeyboardMarkup (class in  
 aiogram.types.reply\_keyboard), 72  
 ReplyKeyboardRemove (class in  
 aiogram.types.reply\_keyboard), 73  
 request() (aiogram.bot.base.BaseBot method), 18  
 request\_timeout() (aiogram.bot.base.BaseBot  
 method), 17  
 required (aiogram.dispatcher.filters.BoundFilter at-  
 tribute), 169  
 reset\_webhook() (aiogram.Dispatcher method),  
 174  
 resolve() (aiogram.dispatcher.filters.FiltersFactory  
 method), 158  
 ResponseParameters (class in  
 aiogram.types.response\_parameters), 86  
 RestartingTelegram, 193  
 restrict() (aiogram.types.chat.Chat method), 75  
 restrict\_chat\_member() (aiogram.bot.bot.Bot  
 method), 39  
 ResultIdDuplicate, 193  
 RethinkDBStorage (class in  
 aiogram.contrib.fsm\_storage.rethinkdb),  
 171  
 RetryAfter, 193  
 row() (aiogram.types.inline\_keyboard.InlineKeyboardMarkup  
 method), 68

`row()` (*aiogram.types.reply\_keyboard.ReplyKeyboardMarkup* method), 72

## S

`safe_split_text()` (in module *aiogram.utils.parts*), 198

`save()` (*aiogram.types.input\_file.InputFile* method), 115

`send_animation()` (*aiogram.bot.bot.Bot* method), 28

`send_audio()` (*aiogram.bot.bot.Bot* method), 25

`send_chat_action()` (*aiogram.bot.bot.Bot* method), 38

`send_contact()` (*aiogram.bot.bot.Bot* method), 35

`send_copy()` (*aiogram.types.message.Message* method), 154

`send_dice()` (*aiogram.bot.bot.Bot* method), 37

`send_document()` (*aiogram.bot.bot.Bot* method), 26

`send_file()` (*aiogram.bot.base.BaseBot* method), 18

`send_game()` (*aiogram.bot.bot.Bot* method), 56

`send_invoice()` (*aiogram.bot.bot.Bot* method), 53

`send_location()` (*aiogram.bot.bot.Bot* method), 31

`send_media_group()` (*aiogram.bot.bot.Bot* method), 31

`send_message()` (*aiogram.bot.bot.Bot* method), 22

`send_photo()` (*aiogram.bot.bot.Bot* method), 24

`send_poll()` (*aiogram.bot.bot.Bot* method), 35

`send_sticker()` (*aiogram.bot.bot.Bot* method), 49

`send_venue()` (*aiogram.bot.bot.Bot* method), 34

`send_video()` (*aiogram.bot.bot.Bot* method), 27

`send_video_note()` (*aiogram.bot.bot.Bot* method), 30

`send_voice()` (*aiogram.bot.bot.Bot* method), 29

`serialize()` (*aiogram.types.fields.BaseField* method), 60

`serialize()` (*aiogram.types.fields.DateTimeField* method), 62

`serialize()` (*aiogram.types.fields.Field* method), 61

`serialize()` (*aiogram.types.fields.ListField* method), 61

`serialize()` (*aiogram.types.fields.ListOfLists* method), 62

`serialize()` (*aiogram.types.fields.TextField* method), 63

`set_administrator_custom_title()` (*aiogram.types.chat.Chat* method), 77

`set_chat_administrator_custom_title()` (*aiogram.bot.bot.Bot* method), 41

`set_chat_description()` (*aiogram.bot.bot.Bot* method), 42

`set_chat_permissions()` (*aiogram.bot.bot.Bot* method), 41

`set_chat_photo()` (*aiogram.bot.bot.Bot* method), 41

`set_chat_sticker_set()` (*aiogram.bot.bot.Bot* method), 45

`set_chat_title()` (*aiogram.bot.bot.Bot* method), 42

`set_description()` (*aiogram.types.chat.Chat* method), 74

`set_game_score()` (*aiogram.bot.bot.Bot* method), 56

`set_my_commands()` (*aiogram.bot.bot.Bot* method), 46

`set_passport_data_errors()` (*aiogram.bot.bot.Bot* method), 55

`set_permissions()` (*aiogram.types.chat.Chat* method), 77

`set_photo()` (*aiogram.types.chat.Chat* method), 74

`set_position_in_set()` (*aiogram.types.sticker.Sticker* method), 87

`set_sticker_position_in_set()` (*aiogram.bot.bot.Bot* method), 51

`set_sticker_set()` (*aiogram.types.chat.Chat* method), 78

`set_sticker_set_thumb()` (*aiogram.bot.bot.Bot* method), 52

`set_title()` (*aiogram.types.chat.Chat* method), 74

`set_value()` (*aiogram.types.fields.BaseField* method), 60

`set_web_app()` (*aiogram.utils.executor.Executor* method), 187

`set_webhook()` (*aiogram.bot.bot.Bot* method), 20

`set_webhook()` (*aiogram.utils.executor.Executor* method), 188

`set_webhook()` (in module *aiogram.utils.executor*), 186

`setup_middleware()` (*aiogram.Dispatcher* method), 184

`shifted_id()` (*aiogram.types.chat.Chat* property), 73

`shipping_query_handler()` (*aiogram.Dispatcher* method), 180

`ShippingAddress` (class in *aiogram.types.shipping\_address*), 86

`ShippingOption` (class in *aiogram.types.shipping\_option*), 124

`ShippingQuery` (class in *aiogram.types.shipping\_query*), 67

`skip_updates()` (*aiogram.Dispatcher* method), 173

`split_text()` (in module *aiogram.utils.parts*), 198

`start()` (*aiogram.utils.executor.Executor* method), 188

`start()` (in module *aiogram.utils.executor*), 187

`start_polling()` (*aiogram.Dispatcher* method), 174

`start_polling()` (*aiogram.utils.executor.Executor* method), 188

`start_polling()` (in module *aiogram.utils.executor*), 186

[start\\_webhook\(\)](#) (*aiogram.utils.executor.Executor method*), 188  
[start\\_webhook\(\)](#) (in module *aiogram.utils.executor*), 186  
[StartParamInvalid](#), 192  
[StateFilter](#) (class in *aiogram.dispatcher.filters*), 166  
[Sticker](#) (class in *aiogram.types.sticker*), 87  
[StickerSet](#) (class in *aiogram.types.sticker\_set*), 64  
[stop\\_live\\_location\(\)](#) (*aiogram.types.message.Message method*), 153  
[stop\\_message\\_live\\_location\(\)](#) (*aiogram.bot.bot.Bot method*), 33  
[stop\\_poll\(\)](#) (*aiogram.bot.bot.Bot method*), 48  
[stop\\_polling\(\)](#) (*aiogram.Dispatcher method*), 174  
[strikethrough\(\)](#) (in module *aiogram.utils.markdown*), 195  
[SuccessfulPayment](#) (class in *aiogram.types.successful\_payment*), 66

## T

[TelegramAPIError](#), 191  
[TelegramAPIServer](#) (class in *aiogram.bot.api*), 57  
[TelegramObject](#) (class in *aiogram.types.base*), 59  
[TerminatedByOtherGetUpdates](#), 193  
[Text](#) (class in *aiogram.dispatcher.filters*), 163  
[text\(\)](#) (*aiogram.dispatcher.filters.Command.CommandObj property*), 160  
[text\(\)](#) (in module *aiogram.utils.markdown*), 194  
[TextField](#) (class in *aiogram.types.fields*), 63  
[throttle\(\)](#) (*aiogram.Dispatcher method*), 183  
[Throttled](#), 193  
[throttled\(\)](#) (*aiogram.Dispatcher method*), 184  
[TimeoutWarning](#), 191  
[to\\_object\(\)](#) (*aiogram.types.base.TelegramObject class method*), 59  
[to\\_object\(\)](#) (*aiogram.types.input\_file.InputFile class method*), 116  
[to\\_python\(\)](#) (*aiogram.types.base.TelegramObject method*), 59  
[to\\_python\(\)](#) (*aiogram.types.input\_file.InputFile method*), 115  
[to\\_python\(\)](#) (*aiogram.types.input\_media.MediaGroup method*), 93  
[TooMuchMessages](#), 191  
[TypeOfFileMismatch](#), 193  
[typing\(\)](#) (*aiogram.types.chat.ChatActions class method*), 80

## U

[Unauthorized](#), 193  
[UnavailableMembers](#), 193  
[unban\(\)](#) (*aiogram.types.chat.Chat method*), 75  
[unban\\_chat\\_member\(\)](#) (*aiogram.bot.bot.Bot method*), 39  
[unbind\(\)](#) (*aiogram.dispatcher.filters.FiltersFactory method*), 158  
[unbind\\_filter\(\)](#) (*aiogram.Dispatcher method*), 184  
[underline\(\)](#) (in module *aiogram.utils.markdown*), 195  
[unpin\(\)](#) (*aiogram.types.message.Message method*), 154  
[unpin\\_all\\_chat\\_messages\(\)](#) (*aiogram.bot.bot.Bot method*), 43  
[unpin\\_all\\_messages\(\)](#) (*aiogram.types.chat.Chat method*), 77  
[unpin\\_chat\\_message\(\)](#) (*aiogram.bot.bot.Bot method*), 43  
[unpin\\_message\(\)](#) (*aiogram.types.chat.Chat method*), 77  
[UnsupportedUrlProtocol](#), 193  
[Update](#) (class in *aiogram.types.update*), 121  
[update\\_chat\(\)](#) (*aiogram.types.chat.Chat method*), 73  
[upload\\_audio\(\)](#) (*aiogram.types.chat.ChatActions class method*), 81  
[upload\\_document\(\)](#) (*aiogram.types.chat.ChatActions class method*), 81  
[upload\\_photo\(\)](#) (*aiogram.types.chat.ChatActions class method*), 81  
[upload\\_sticker\\_file\(\)](#) (*aiogram.bot.bot.Bot method*), 50  
[upload\\_video\(\)](#) (*aiogram.types.chat.ChatActions class method*), 81  
[upload\\_video\\_note\(\)](#) (*aiogram.types.chat.ChatActions class method*), 81  
[url\(\)](#) (*aiogram.types.message.Message property*), 126  
[URLHostIsEmpty](#), 192  
[User](#) (class in *aiogram.types.user*), 69  
[UserDeactivated](#), 193  
[UserProfilePhotos](#) (class in *aiogram.types.user\_profile\_photos*), 157

## V

[validate\(\)](#) (*aiogram.dispatcher.filters.AbstractFilter class method*), 168  
[validate\(\)](#) (*aiogram.dispatcher.filters.AdminFilter class method*), 167  
[validate\(\)](#) (*aiogram.dispatcher.filters.BoundFilter class method*), 169  
[validate\(\)](#) (*aiogram.dispatcher.filters.builtin.IDFilter class method*), 166  
[validate\(\)](#) (*aiogram.dispatcher.filters.Command class method*), 160

`validate()` (*aiogram.dispatcher.filters.Filter* class method), [169](#)  
`validate()` (*aiogram.dispatcher.filters.HashTag* class method), [164](#)  
`validate()` (*aiogram.dispatcher.filters.Regexp* class method), [164](#)  
`validate()` (*aiogram.dispatcher.filters.Text* class method), [163](#)  
`ValidationError`, [191](#)  
`values()` (*aiogram.types.base.TelegramObject* property), [59](#)  
`Venue` (class in *aiogram.types.venue*), [122](#)  
`Video` (class in *aiogram.types.video*), [70](#)  
`VideoNote` (class in *aiogram.types.video\_note*), [122](#)  
`Voice` (class in *aiogram.types.voice*), [116](#)

## W

`wait_closed()` (*aiogram.Dispatcher* method), [174](#)  
`WebhookInfo` (class in *aiogram.types.webhook\_info*), [123](#)  
`WebhookRequireHTTPS`, [193](#)  
`WrongFileIdentifier`, [192](#)  
`WrongRemoteFileIdSpecified`, [193](#)