# aiogram Documentation

Release 3.13.1

aiogram Team

## **CONTENTS**

1 Features						
	1.1	Simple	usage			
	1.2	Usage v	vithout dispatcher			
2	Conte	ents	7			
	2.1	Installat	tion			
		2.1.1	From PyPI			
		2.1.2	From Arch Linux Repository			
		2.1.3	From GitHub			
	2.2	Migrati	on FAQ (2.x -> 3.0)			
		2.2.1	Dependencies			
		2.2.2	Dispatcher			
		2.2.3	Filtering events			
		2.2.4	Bot API			
		2.2.5	Middlewares			
		2.2.6	Keyboard Markup			
		2.2.7	Callbacks data			
		2.2.8	Finite State machine			
		2.2.9	Sending Files			
		2.2.10	Webhook			
		2.2.11	Telegram API Server			
		2.2.12	Telegram objects transformation (to dict, to json, from json)			
		2.2.13	ChatMember tools			
	2.3		I			
	2.0	2.3.1	Bot			
		2.3.2	Client session			
		2.3.3	Types			
		2.3.4	Methods			
		2.3.5	Enums			
		2.3.6	How to download file?			
		2.3.7	How to upload file?			
		2.3.8	Global defaults			
	2.4		ag events			
	2.7	2.4.1	Router			
		2.4.2	Dispatcher			
		2.4.3	Dependency injection			
		2.4.3	Filtering events			
		2.4.4	Long-polling			
		2.4.5	Finite State Machine			
		2.4.0				
		∠.4./	Middlewares			

	2.4.8	Errors
	2.4.9	Flags
	2.4.10	Webhook
	2.4.10	Class based handlers
2.5		
2.5		
	2.5.1	Keyboard builder
	2.5.2	Translation
	2.5.3	Chat action sender
	2.5.4	WebApp
	2.5.5	Callback answer
	2.5.6	Formatting
	2.5.7	Media group builder
	2.5.8	Deep Linking
	2.5.9	Telegram object serialization
2.6	Change	log
	2.6.1	3.13.1 (2024-09-18)
	2.6.2	3.13.0 (2024-09-08)
	2.6.3	3.12.0 (2024-08-16)
	2.6.4	3.11.0 (2024-08-09)
	2.6.5	3.10.0 (2024-07-07)
	2.6.6	3.9.0 (2024-07-06)
	2.6.7	3.8.0 (2024-06-19)
	2.6.8	3.7.0 (2024-05-31)
	2.6.9	3.6.0 (2024-05-06)
	2.6.10	3.5.0 (2024-04-23)
	2.6.11	3.4.1 (2024-02-17)
	2.6.12	3.4.0 (2024-02-16)
	2.6.13	3.3.0 (2023-12-31)
	2.6.14	3.2.0 (2023-11-24)
	2.6.15	3.1.1 (2023-09-25)
	2.6.16	3.1.0 (2023-09-22)
	2.6.17	3.0.0 (2023-09-01)
	2.6.18	3.0.0rc2 (2023-08-18)
	2.6.19	3.0.0rc1 (2023-08-06)
	2.6.20	3.0.0b9 (2023-07-30)
	2.6.21	3.0.068 (2023-07-17)
	2.6.22	3.0.067 (2023-02-18)
	2.6.23	3.0.0b6 (2022-11-18)
	2.6.24	3.0.0b5 (2022-10-02)
	2.6.25	3.0.0b4 (2022-08-14)
	2.6.26	3.0.0b3 (2022-04-19)
	2.6.27	3.0.0b2 (2022-02-19)
	2.6.28	3.0.0b1 (2021-12-12)
	2.6.29	3.0.0a18 (2021-12-12)
		3.0.0a17 (2021-09-24)
	2.6.30	
	2.6.31	3.0.0a16 (2021-09-22)
	2.6.32	3.0.0a15 (2021-09-10)
	2.6.33	3.0.0a14 (2021-08-17)
	2.6.34	2.14.3 (2021-07-21)
	2.6.35	2.14.2 (2021-07-26)
	2.6.36	2.14 (2021-07-27)
	2.6.37	2.13 (2021-04-28)
	2.6.38	2.12.1 (2021-03-22)
	2.6.39	2.12 (2021-03-14)

	2.6.40 2.6.41 2.6.42 2.6.43 2.6.44 2.6.45 2.6.46 2.6.47 2.6.48 2.6.50 2.6.51 2.6.52 2.6.53 2.6.54 2.6.55 2.6.56 2.6.57 2.6.58 2.6.60 2.6.61 2.6.62 2.6.63 2.6.64 2.6.65 2.6.65 2.6.67 2.6.68 2.6.70 2.6.71 2.6.72 2.6.73 2.6.74 2.6.75 2.6.76 2.6.77	2.11.2 (2021-11-10) 2.11.1 (2021-11-10) 2.11.1 (2021-11-08) 2.10.1 (2021-09-14) 2.10.1 (2021-09-13) 2.9.2 (2021-06-08) 2.8 (2021-06-08) 2.8 (2021-04-26) 2.7 (2021-04-07) 2.6.1 (2021-01-25) 2.6.1 (2021-01-23) 2.5.3 (2021-01-05) 2.5.2 (2021-01-01) 2.5.1 (2021-01-01) 2.5.1 (2021-01-01) 2.5.1 (2021-01-01) 2.4 (2021-029) 2.3 (2021-08-16) 2.2 (2021-06-09) 2.1 (2021-01-28) 1.4 (2021-028) 1.4 (2021-08-03) 1.3 (2021-07-16) 1.3.2 (2021-07-17) 1.3.1 (2018-05-27) 1.3.1 (2018-05-27) 1.3.1 (2018-05-27) 1.3.1 (2018-07-27) 1.3.1 (2018-07-28) 1.1 (2018-07-28) 1.1 (2018-07-29) 1.1 (2018-07-29) 1.2 (2018-07-16) 1.3 (2021-07-16) 1.3 (2021-07-16) 1.3 (2021-07-16) 1.3 (2021-07-16) 1.3 (2021-07-16) 1.3 (2021-07-16) 1.3 (2021-07-16) 1.3 (2021-07-16) 1.3 (2021-07-16) 1.3 (2021-07-16) 1.3 (2021-07-16) 1.3 (2021-07-16) 1.3 (2021-07-16) 1.3 (2021-07-07-16) 1.3 (2018-07-17) 1.0 (2017-07-08) 1.1 (2017-08-03) 0.4 (2017-08-03) 0.4 (2017-08-03) 0.3 (2017-07-04) 0.3 (2017-07-04) 0.3 (2017-07-04) 0.3 (2017-07-04) 0.3 (2017-07-04) 0.3 (2017-07-04) 0.3 (2017-07-04) 0.3 (2017-07-04) 0.3 (2017-07-04) 0.3 (2017-07-04) 0.3 (2017-07-04) 0.3 (2017-07-04) 0.3 (2017-07-04) 0.3 (2017-07-04) 0.3 (2017-07-04)	. 701 . 701 . 701 . 701 . 702 . 702 . 703 . 703 . 703 . 704 . 704 . 704 . 705 . 705 . 705 . 706 . 706 . 706 . 706 . 707 . 707 . 707 . 707 . 707 . 707 . 707 . 707 . 708 . 708 . 708 . 708 . 708 . 708 . 708 . 709 . 709 . 709 . 709 . 709
	2.6.81	0.3.1 (2017-07-04)	. 709
2.5			
2.7		uting	
	2.7.1 2.7.2	Developing	
		Star on GitHub	
	2.7.3	Guides	
	2.7.4	Take answers	
	2.7.5	Funding	. 712
Python N Index	Module I	index	713 719

 ${f aiogram}$  is a modern and fully asynchronous framework for Telegram Bot API written in Python 3.8+ using asyncio and aiohttp.

Make your bots faster and more powerful!

### **Documentation:**

- English
- Ukrainian

CONTENTS 1

2 CONTENTS

#### **CHAPTER**

### ONE

### **FEATURES**

- Asynchronous (asyncio docs, PEP 492)
- Has type hints (PEP 484) and can be used with mypy
- Supports PyPy
- Supports Telegram Bot API 7.10 and gets fast updates to the latest versions of the Bot API
- Telegram Bot API integration code was autogenerated and can be easily re-generated when API gets updated
- Updates router (Blueprints)
- Has Finite State Machine
- Uses powerful magic filters
- Middlewares (incoming updates and API calls)
- Provides Replies into Webhook
- Integrated I18n/L10n support with GNU Gettext (or Fluent)

### **▲** Warning

It is strongly advised that you have prior experience working with asyncio before beginning to use aiogram.

If you have any questions, you can visit our community chats on Telegram:

- @aiogram
- @aiogramua
- @aiogram\_uz
- @aiogram\_kz
- @aiogram\_ru
- @aiogram\_fa
- @aiogram\_it
- @aiogram\_br

### 1.1 Simple usage

```
import asyncio
import logging
import sys
from os import getenv
from aiogram import Bot, Dispatcher, html
from aiogram.client.default import DefaultBotProperties
from aiogram.enums import ParseMode
from aiogram.filters import CommandStart
from aiogram.types import Message
# Bot token can be obtained via https://t.me/BotFather
TOKEN = getenv("BOT_TOKEN")
# All handlers should be attached to the Router (or Dispatcher)
dp = Dispatcher()
@dp.message(CommandStart())
async def command_start_handler(message: Message) -> None:
    This handler receives messages with `/start` command
   # Most event objects have aliases for API methods that can be called in events.
   # For example if you want to answer to incoming message you can use `message.answer(.
→...)` alias
   # and the target chat will be passed to :ref:`aiogram.methods.send_message.
→ SendMessage`
   # method automatically or call API method directly via
    # Bot instance: `bot.send_message(chat_id=message.chat.id, ...)`
   await message.answer(f"Hello, {html.bold(message.from_user.full_name)}!")
@dp.message()
async def echo_handler(message: Message) -> None:
   Handler will forward receive a message back to the sender
   By default, message handler will handle all message types (like a text, photo,
⇒sticker etc.)
    0.00
   trv:
        # Send a copy of the received message
        await message.send_copy(chat_id=message.chat.id)
   except TypeError:
        # But not all the types is supported to be copied so need to handle it
        await message.answer("Nice try!")
```

(continues on next page)

(continued from previous page)

```
async def main() -> None:
    # Initialize Bot instance with default bot properties which will be passed to all_
    API calls
    bot = Bot(token=TOKEN, default=DefaultBotProperties(parse_mode=ParseMode.HTML))

# And the run events dispatching
    await dp.start_polling(bot)

if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, stream=sys.stdout)
    asyncio.run(main())
```

### 1.2 Usage without dispatcher

Just only interact with Bot API, without handling events

```
import asyncio
from argparse import ArgumentParser
from aiogram import Bot
from aiogram.client.default import DefaultBotProperties
from aiogram.enums import ParseMode
def create_parser() -> ArgumentParser:
   parser = ArgumentParser()
   parser.add_argument("--token", help="Telegram Bot API Token")
   parser.add_argument("--chat-id", type=int, help="Target chat id")
   parser.add_argument("--message", "-m", help="Message text to sent", default="Hello,_
→World!")
   return parser
async def main():
   parser = create_parser()
   ns = parser.parse_args()
   token = ns.token
   chat_id = ns.chat_id
   message = ns.message
   async with Bot(
        token=token,
        default=DefaultBotProperties(
           parse_mode=ParseMode.HTML,
        ),
   ) as bot:
```

(continues on next page)

(continued from previous page)

```
await bot.send_message(chat_id=chat_id, text=message)

if __name__ == "__main__":
    asyncio.run(main())
```

6 Chapter 1. Features

**CHAPTER** 

**TWO** 

### **CONTENTS**

### 2.1 Installation

### 2.1.1 From PyPI

pip install -U aiogram

### 2.1.2 From Arch Linux Repository



#### Warning

Package in this repository may be outdated. Use PyPI package for the latest version.

pacman -S python-aiogram

#### **Development build**

### 2.1.3 From GitHub

pip install https://github.com/aiogram/aiogram/archive/refs/heads/dev-3.x.zip

### 2.2 Migration FAQ $(2.x \rightarrow 3.0)$



This guide is still in progress.

This version introduces numerous breaking changes and architectural improvements. It helps reduce the count of global variables in your code, provides useful mechanisms to modularize your code, and enables the creation of shareable modules via packages on PyPI. It also makes middlewares and filters more controllable, among other improvements.

On this page, you can read about the changes made in relation to the last stable 2.x version.

#### 1 Note

This page more closely resembles a detailed changelog than a migration guide, but it will be updated in the future.

Feel free to contribute to this page, if you find something that is not mentioned here.

### 2.2.1 Dependencies

• The dependencies required for i18n are no longer part of the default package. If your application uses translation functionality, be sure to add an optional dependency:

pip install aiogram[i18n]

### 2.2.2 Dispatcher

- The Dispatcher class no longer accepts a Bot instance in its initializer. Instead, the Bot instance should be passed to the dispatcher only for starting polling or handling events from webhooks. This approach also allows for the use of multiple bot instances simultaneously ("multibot").
- Dispatcher now can be extended with another Dispatcher-like thing named Router. With routes, you can easily modularize your code and potentially share these modules between projects. (Read more ».)
- Removed the \_handler suffix from all event handler decorators and registering methods. (Read more »)
- The Executor has been entirely removed; you can now use the Dispatcher directly to start poll the API or handle webhooks from it.
- The throttling method has been completely removed; you can now use middlewares to control the execution context and implement any throttling mechanism you desire.
- Removed global context variables from the API types, Bot and Dispatcher object. From now on, if you want to access the current bot instance within handlers or filters, you should accept the argument bot: Bot and use it instead of Bot.get\_current(). In middlewares, it can be accessed via data["bot"].
- To skip pending updates, you should now call the DeleteWebhook method directly, rather than passing skip\_updates=True to the start polling method.
- To feed updates to the Dispatcher, instead of method process\_update(), you should use method feed\_update(). (Read more »)

### 2.2.3 Filtering events

- Keyword filters can no longer be used; use filters explicitly. (Read more »)
- Due to the removal of keyword filters, all previously enabled-by-default filters (such as state and content\_type) are now disabled. You must specify them explicitly if you wish to use them. For example instead of using @dp. message\_handler(content\_types=ContentType.PHOTO) you should use @router.message(F.photo)
- Most common filters have been replaced with the "magic filter." (Read more »)
- By default, the message handler now receives any content type. If you want a specific one, simply add the appropriate filters (Magic or any other).
- The state filter is no longer enabled by default. This means that if you used state="\*" in v2, you should not pass any state filter in v3. Conversely, if the state was not specified in v2, you will now need to specify it in v3.

 Added the possibility to register global filters for each router, which helps to reduce code repetition and provides an easier way to control the purpose of each router.

#### 2.2.4 Bot API

- All API methods are now classes with validation, implemented via *pydantic < https://docs.pydantic.dev/>*. These API calls are also available as methods in the Bot class.
- More pre-defined Enums have been added and moved to the *aiogram.enums* sub-package. For example, the chat type enum is now aiogram.enums.ChatType instead of aiogram.types.chat.ChatType.
- The HTTP client session has been separated into a container that can be reused across different Bot instances within the application.
- API Exceptions are no longer classified by specific messages, as Telegram has no documented error codes.
  However, all errors are classified by HTTP status codes, and for each method, only one type of error can be
  associated with a given code. Therefore, in most cases, you should check only the error type (by status code)
  without inspecting the error message.

#### 2.2.5 Middlewares

- Middlewares can now control an execution context, e.g., using context managers. (Read more »)
- All contextual data is now shared end-to-end between middlewares, filters, and handlers. For example now you
  can easily pass some data into context inside middleware and get it in the filters layer as the same way as in the
  handlers via keyword arguments.
- Added a mechanism named flags that helps customize handler behavior in conjunction with middlewares. (Read more »)

### 2.2.6 Keyboard Markup

• Now aiogram.types.inline\_keyboard\_markup.InlineKeyboardMarkup and aiogram.types. reply\_keyboard\_markup.ReplyKeyboardMarkup no longer have methods for extension, instead you have to use markup builders aiogram.utils.keyboard.ReplyKeyboardBuilder and aiogram.utils.keyboard.KeyboardBuilder respectively (Read more »)

#### 2.2.7 Callbacks data

• The callback data factory is now strictly typed using pydantic models. (*Read more* »)

#### 2.2.8 Finite State machine

- State filters will no longer be automatically added to all handlers; you will need to specify the state if you want to use it.
- Added the possibility to change the FSM strategy. For example, if you want to control the state for each user based on chat topics rather than the user in a chat, you can specify this in the *Dispatcher*.
- Now aiogram.fsm.state.State and aiogram.fsm.state.StateGroup don't have helper methods like .set(), .next(), etc. Instead, you should set states by passing them directly to aiogram.fsm.context. FSMContext (Read more »)

• The state proxy is deprecated; you should update the state data by calling state.set\_data(...) and state. get\_data() respectively.

### 2.2.9 Sending Files

• From now on, you should wrap files in an InputFile object before sending them, instead of passing the IO object directly to the API method. (*Read more* »)

#### 2.2.10 Webhook

- The aiohttp web app configuration has been simplified.
- By default, the ability to upload files has been added when you make requests in response to updates (available for webhook only).

### 2.2.11 Telegram API Server

- The server parameter has been moved from the Bot instance to api parameter of the BaseSession.
- The constant aiogram.bot.api.TELEGRAM\_PRODUCTION has been moved to aiogram.client.telegram. PRODUCTION.

### 2.2.12 Telegram objects transformation (to dict, to json, from json)

- Methods TelegramObject.to\_object(), TelegramObject.to\_json() and TelegramObject.to\_python() have been removed due to the use of pydantic models.
- TelegramObject.to\_object() should be replaced by TelegramObject.model\_validate() (Read more)
- TelegramObject.as\_json() should be replaced by aiogram.utils.serialization. deserialize\_telegram\_object\_to\_python()
- <TelegramObject>.to\_python() should be replaced by json.dumps(deserialize\_telegram\_object\_to\_python(<Tele

Here are some usage examples:

• Creating an object from a dictionary representation of an object

```
# Version 2.x
message_dict = {"id": 42, ...}
message_obj = Message.to_object(message_dict)
print(message_obj)
# id=42 name='n' ...
print(type(message_obj))
# <class 'aiogram.types.message.Message'>
```

```
# Version 3.x
message_dict = {"id": 42, ...}
message_obj = Message.model_validate(message_dict)
print(message_obj)
# id=42 name='n' ...
print(type(message_obj))
# <class 'aiogram.types.message.Message'>
```

• Creating a json representation of an object

```
# Version 2.x
async def handler(message: Message) -> None:
    message_json = message.as_json()
    print(message_json)
    # {"id": 42, ...}
    print(type(message_json))
    # <class 'str'>
```

```
# Version 3.x
async def handler(message: Message) -> None:
    message_json = json.dumps(deserialize_telegram_object_to_python(message))
    print(message_json)
    # {"id": 42, ...}
    print(type(message_json))
    # <class 'str'>
```

· Creating a dictionary representation of an object

```
async def handler(message: Message) -> None:
    # Version 2.x
    message_dict = message.to_python()
    print(message_dict)
    # {"id": 42, ...}
    print(type(message_dict))
    # <class 'dict'>
```

```
async def handler(message: Message) -> None:
    # Version 3.x
    message_dict = deserialize_telegram_object_to_python(message)
    print(message_dict)
    # {"id": 42, ...}
    print(type(message_dict))
    # <class 'dict'>
```

#### 2.2.13 ChatMember tools

• Now aiogram.types.chat\_member.ChatMember no longer contains tools to resolve an object with the appropriate status.

```
# Version 2.x
from aiogram.types import ChatMember
chat_member = ChatMember.resolve(**dict_data)
```

```
# Version 3.x
from aiogram.utils.chat_member import ChatMemberAdapter
chat_member = ChatMemberAdapter.validate_python(dict_data)
```

• Now aiogram.types.chat\_member.ChatMember and all its child classes no longer contain methods for

checking for membership in certain logical groups. As a substitute, you can use pre-defined groups or create such groups yourself and check their entry using the isinstance() function

```
# Version 2.x

if chat_member.is_chat_admin():
    print("ChatMember is chat admin")

if chat_member.is_chat_member():
    print("ChatMember is in the chat")
```

```
# Version 3.x
from aiogram.utils.chat_member import ADMINS, MEMBERS

if isinstance(chat_member, ADMINS):
    print("ChatMember is chat admin")

if isinstance(chat_member, MEMBERS):
    print("ChatMember is in the chat")
```

#### **1** Note

You also can independently create group similar to ADMINS that fits the logic of your application.

E.g., you can create a PUNISHED group and include banned and restricted members there!

### 2.3 Bot API

aiogram now is fully support of Telegram Bot API

All methods and types is fully autogenerated from Telegram Bot API docs by parser with code-generator.

#### 2.3.1 Bot

Bot instance can be created from aiogram.Bot (from aiogram import Bot) and you can't use methods without instance of bot with configured token.

This class has aliases for all methods and named in lower\_camel\_case.

For example sendMessage named send\_message and has the same specification with all class-based methods.

#### **A** Warning

A full list of methods can be found in the appropriate section of the documentation

Bases: object

```
__init__(token: str, session: BaseSession | None = None, default: DefaultBotProperties | None = None, **kwargs: Any) → None
```

Bot class

#### **Parameters**

- token Telegram Bot token Obtained from @BotFather
- **session** HTTP Client session (For example AiohttpSession). If not specified it will be automatically created.
- default Default bot properties. If specified it will be propagated into the API methods at runtime.

#### **Raises**

TokenValidationError – When token has invalid format this exception will be raised

```
property token: str
```

property id: int

Get bot ID from token

#### **Returns**

**context**( $auto\_close: bool = True$ )  $\rightarrow$  AsyncIterator[Bot]

Generate bot context

#### **Parameters**

auto\_close - close session on exit

#### Returns

```
async me() \rightarrow User
```

Cached alias for getMe method

#### Returns

```
async download_file(file_path: str, destination: BinaryIO | Path | str | None = None, timeout: int = 30, chunk size: int = 65536, seek: bool = True) \rightarrow BinaryIO | None
```

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** File path on Telegram server (You can get it from aiogram.types.File)
- **destination** Filename, file path or instance of io.IOBase. For e.g. io.BytesIO, defaults to None
- timeout Total timeout in seconds, defaults to 30
- chunk\_size File chunks size, defaults to 64 kb
- **seek** Go to start of file when downloading is finished. Used only for destination with typing.BinaryIO type, defaults to True

**async download**(file:  $str \mid Downloadable$ , destination:  $BinaryIO \mid Path \mid str \mid None = None$ , timeout: int = 30,  $chunk\_size$ : int = 65536, seek:  $bool = True) \rightarrow BinaryIO \mid None$ 

Download file by file id or Downloadable object to destination.

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

#### **Parameters**

- file file\_id or Downloadable object
- **destination** Filename, file path or instance of io.IOBase. For e.g. io.BytesIO, defaults to None
- timeout Total timeout in seconds, defaults to 30
- **chunk\_size** File chunks size, defaults to 64 kb
- **seek** Go to start of file when downloading is finished. Used only for destination with typing.BinaryIO type, defaults to True

#### 2.3.2 Client session

Client sessions is used for interacting with API server.

#### **Use Custom API server**

For example, if you want to use self-hosted API server:

• token - Bot token

```
session = AiohttpSession(
    api=TelegramAPIServer.from_base('http://localhost:8082')
bot = Bot(..., session=session)
class aiogram.client.telegram.TelegramAPIServer(base: str, file: str, is_local: bool = False,
                                                         wrap_local_file:
                                                         ~aiogram.client.telegram.FilesPathWrapper =
                                                         <aiogram.client.telegram.BareFilesPathWrapper
                                                         object>)
     Base config for API Endpoints
     api\_url(token: str, method: str) \rightarrow str
          Generate URL for API methods
              Parameters
                   • token - Bot token
                   • method – API method name (case insensitive)
              Returns
                  URL
     base: str
          Base URL
     file: str
          Files URL
     file_url(token: str, path: str) \rightarrow str
          Generate URL for downloading files
              Parameters
```

```
• path - file path
               Returns
                   URL
     classmethod from_base(base: str, **kwargs: Any) \rightarrow TelegramAPIServer
           Use this method to auto-generate TelegramAPIServer instance from base URL
               Parameters
                   base – Base URL
               Returns
                   instance of TelegramAPIServer
     is_local: bool = False
           Mark this server is in local mode.
     wrap_local_file: FilesPathWrapper = <aiogram.client.telegram.BareFilesPathWrapper</pre>
     object>
           Callback to wrap files path in local mode
Base
Abstract session for all client sessions
class aiogram.client.session.base.BaseSession(api: ~aiogram.client.telegram.TelegramAPIServer =
                                                          TelegramAPIS-
                                                          erver(base='https://api.telegram.org/bot{token}/{method}',
                                                         file='https://api.telegram.org/file/bot{token}/{path}',
                                                          is_local=False,
                                                          wrap local file=<aiogram.client.telegram.BareFilesPathWrapper
                                                          object>), json_loads: ~typing.Callable[[...],
                                                          \sim typing.Any] = < function loads>, json_dumps:
                                                          \sim typing.Callable[[...], str] = < function dumps>,
                                                          timeout: float = 60.0)
     This is base class for all HTTP sessions in aiogram.
     If you want to create your own session, you must inherit from this class.
     \textbf{check\_response}(bot: Bot, method: TelegramMethod[TelegramType], status\_code: int, content: str) \rightarrow
                        Response[TelegramType]
           Check response status
     abstract async close() \rightarrow None
           Close client session
     abstract async make_request(bot: Bot, method: TelegramMethod[TelegramType], timeout: int | None =
                                         None) \rightarrow TelegramType
           Make request to Telegram Bot API
               Parameters
                    • bot – Bot instance

    method – Method instance

                    • timeout – Request timeout
```

2.3. Bot API 15

Returns

#### **Raises**

TelegramApiError -

Stream reader

#### aiohttp

AiohttpSession represents a wrapper-class around ClientSession from aiohttp

Currently AiohttpSession is a default session used in aiogram.Bot

#### Usage example

```
from aiogram import Bot
from aiogram.client.session.aiohttp import AiohttpSession

session = AiohttpSession()
bot = Bot('42:token', session=session)
```

#### **Proxy requests in AiohttpSession**

In order to use AiohttpSession with proxy connector you have to install aiohttp-socks

Binding session to bot:

```
from aiogram import Bot
from aiogram.client.session.aiohttp import AiohttpSession

session = AiohttpSession(proxy="protocol://host:port/")
bot = Bot(token="bot token", session=session)
```

#### 1 Note

Only following protocols are supported: http(tunneling), socks4(a), socks5 as aiohttp\_socks documentation claims.

#### **Authorization**

Proxy authorization credentials can be specified in proxy URL or come as an instance of aiohttp.BasicAuth containing login and password.

Consider examples:

```
from aiohttp import BasicAuth
from aiogram.client.session.aiohttp import AiohttpSession

auth = BasicAuth(login="user", password="password")
session = AiohttpSession(proxy=("protocol://host:port", auth))
```

or simply include your basic auth credential in URL

```
session = AiohttpSession(proxy="protocol://user:password@host:port")
```



Aiogram prefers *BasicAuth* over username and password in URL, so if proxy URL contains login and password and *BasicAuth* object is passed at the same time aiogram will use login and password from *BasicAuth* instance.

#### **Proxy chains**

Since aiohttp-socks supports proxy chains, you're able to use them in aiogram

Example of chain proxies:

```
from aiohttp import BasicAuth
from aiogram.client.session.aiohttp import AiohttpSession

auth = BasicAuth(login="user", password="password")
session = AiohttpSession(
    proxy={
        "protocol0://host0:port0",
        "protocol1://user:password@host1:port1",
        ("protocol2://host2:port2", auth),
    } # can be any iterable if not set
)
```

#### Client session middlewares

In some cases you may want to add some middlewares to the client session to customize the behavior of the client.

Some useful cases that is:

- Log the outgoing requests
- Customize the request parameters
- Handle rate limiting errors and retry the request
- others ...

So, you can do it using client session middlewares. A client session middleware is a function (or callable class) that receives the request and the next middleware to call. The middleware can modify the request and then call the next middleware to continue the request processing.

#### How to register client session middleware?

#### Register using register method

```
bot.session.middleware(RequestLogging(ignore_methods=[GetUpdates]))
```

#### Register using decorator

```
@bot.session.middleware()
async def my_middleware(
    make_request: NextRequestMiddlewareType[TelegramType],
    bot: "Bot",
    method: TelegramMethod[TelegramType],
) -> Response[TelegramType]:
    # do something with request
    return await make_request(bot, method)
```

#### **Example**

#### Class based session middleware

```
class RequestLogging(BaseRequestMiddleware):
       def __init__(self, ignore_methods: Optional[List[Type[TelegramMethod[Any]]]] = None):
           Middleware for logging outgoing requests
           :param ignore_methods: methods to ignore in logging middleware
6
           self.ignore_methods = ignore_methods if ignore_methods else []
       async def __call__(
10
           self,
           make_request: NextRequestMiddlewareType[TelegramType],
12
           bot: "Bot",
13
           method: TelegramMethod[TelegramType],
14
       ) -> Response[TelegramType]:
           if type(method) not in self.ignore_methods:
16
               loggers.middlewares.info(
                    "Make request with method=%r by bot id=%d",
18
                    type(method).__name__,
19
                    bot.id,
20
               )
21
           return await make_request(bot, method)
22
```



this middleware is already implemented inside aiogram, so, if you want to use it you can just import it from aiogram.client.session.middlewares.request\_logging import RequestLogging

#### **Function based session middleware**

```
async def __call__(
    self,
    make_request: NextRequestMiddlewareType[TelegramType],
    bot: "Bot",
    method: TelegramMethod[TelegramType],
) -> Response[TelegramType]:
    try:
        # do something with request
        return await make_request(bot, method)
    finally:
        # do something after request
```

### **2.3.3 Types**

Here is list of all available API types:

#### **Available types**

#### **Animation**

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

Source: https://core.telegram.org/bots/api#animation

```
file_id: str
```

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

```
file_unique_id: str
```

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

width: int

Video width as defined by the sender

height: int

Video height as defined by the sender

duration: int

Duration of the video in seconds as defined by the sender

### model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### thumbnail: *PhotoSize* | None

Optional. Animation thumbnail as defined by the sender

#### file\_name: str | None

Optional. Original animation filename as defined by the sender

#### mime\_type: str | None

Optional. MIME type of the file as defined by the sender

#### file\_size: int | None

*Optional*. File size in bytes. It can be bigger than 2<sup>3</sup>1 and some programming languages may have difficulty/silent defects in interpreting it. But it has at most 52 significant bits, so a signed 64-bit integer or double-precision float type are safe for storing this value.

#### **Audio**

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

Source: https://core.telegram.org/bots/api#audio

#### file\_id: str

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

#### file\_unique\_id: str

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

#### duration: int

Duration of the audio in seconds as defined by the sender

#### performer: str | None

Optional. Performer of the audio as defined by the sender or by audio tags

#### title: str | None

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

### model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### file\_name: str | None

20

Optional. Original filename as defined by the sender

```
mime_type: str | None
```

Optional. MIME type of the file as defined by the sender

```
file_size: int | None
```

*Optional.* File size in bytes. It can be bigger than 2<sup>31</sup> and some programming languages may have difficulty/silent defects in interpreting it. But it has at most 52 significant bits, so a signed 64-bit integer or double-precision float type are safe for storing this value.

```
thumbnail: PhotoSize | None
```

Optional. Thumbnail of the album cover to which the music file belongs

#### BackgroundFill

```
class aiogram.types.background_fill.BackgroundFill(**extra_data: Any)
```

This object describes the way a background is filled based on the selected colors. Currently, it can be one of

- aiogram.types.background\_fill\_solid.BackgroundFillSolid
- aiogram.types.background\_fill\_gradient.BackgroundFillGradient
- $\bullet \ \ aiogram.types.background\_fill\_freeform\_gradient.BackgroundFillFreeformGradient$

Source: https://core.telegram.org/bots/api#backgroundfill

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### BackgroundFillFreeformGradient

```
class aiogram.types.background_fill_freeform_gradient.BackgroundFillFreeformGradient(*,
```

```
type:
Lit-
eral['freeform_gradient
=
'freeform_gradient',
col-
ors:
List[int],
**ex-
tra_data:
Any)
```

The background is a freeform gradient that rotates after every message in the chat.

Source: https://core.telegram.org/bots/api#backgroundfillfreeformgradient

```
type: Literal['freeform_gradient']
```

Type of the background fill, always 'freeform\_gradient'

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
colors: List[int]
```

A list of the 3 or 4 base colors that are used to generate the freeform gradient in the RGB24 format

#### BackgroundFillGradient

The background is a gradient fill.

Source: https://core.telegram.org/bots/api#backgroundfillgradient

```
type: Literal['gradient']
```

Type of the background fill, always 'gradient'

top\_color: int

Top color of the gradient in the RGB24 format

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

bottom\_color: int

Bottom color of the gradient in the RGB24 format

rotation\_angle: int

Clockwise rotation angle of the background fill in degrees; 0-359

### BackgroundFillSolid

The background is filled using the selected color.

Source: https://core.telegram.org/bots/api#backgroundfillsolid

```
type: Literal['solid']
```

Type of the background fill, always 'solid'

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

color: int

The color of the background fill in the RGB24 format

#### BackgroundType

```
class aiogram.types.background_type.BackgroundType(**extra_data: Any)
```

This object describes the type of a background. Currently, it can be one of

- aiogram.types.background\_type\_fill.BackgroundTypeFill
- aiogram.types.background\_type\_wallpaper.BackgroundTypeWallpaper
- aiogram.types.background\_type\_pattern.BackgroundTypePattern
- aiogram.types.background\_type\_chat\_theme.BackgroundTypeChatTheme

Source: https://core.telegram.org/bots/api#backgroundtype

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### **BackgroundTypeChatTheme**

The background is taken directly from a built-in chat theme.

Source: https://core.telegram.org/bots/api#backgroundtypechattheme

```
type: Literal['chat_theme']
```

Type of the background, always 'chat\_theme'

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
theme_name: str
```

Name of the chat theme, which is usually an emoji

### BackgroundTypeFill

The background is automatically filled based on the selected colors.

 $Source: \ https://core.telegram.org/bots/api\#backgroundtype fill$ 

```
type: Literal['fill']
```

Type of the background, always 'fill'

 ${\bf fill:} \ \ \textit{BackgroundFillSolid} \ \ | \ \ \textit{BackgroundFillGradient} \ \ | \ \ \textit{BackgroundFillFreeformGradient}$ 

The background fill

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### dark\_theme\_dimming: int

Dimming of the background in dark themes, as a percentage; 0-100

#### BackgroundTypePattern

 $\textbf{class} \ \texttt{aiogram.types.background\_type\_pattern.BackgroundTypePattern}(*, \textit{type: Literal['pattern']} = \texttt{aiogram.types.background\_type\_pattern}(*, \textit{type: Literal['pattern']} = \texttt{aiogram.types.background\_type\_pattern}(*, \textit{type: Literal['pattern']} = \texttt{aiogram.type\_pattern}(*, \textit{type: Literal['pattern']} = \texttt{aiogram.type\_pattern'} = \texttt{aiogram.typ$ 

'pattern', document:
Document, fill:
BackgroundFillSolid |

BackgroundFillGradient | BackgroundFillFreeformGra-

dient, intensity: int,

is\_inverted: bool | None =
None, is\_moving: bool | None
= None, \*\*extra\_data: Any)

The background is a PNG or TGV (gzipped subset of SVG with MIME type 'application/x-tgwallpattern') pattern to be combined with the background fill chosen by the user.

Source: https://core.telegram.org/bots/api#backgroundtypepattern

#### type: Literal['pattern']

Type of the background, always 'pattern'

#### document: Document

Document with the pattern

fill: BackgroundFillSolid | BackgroundFillGradient | BackgroundFillFreeformGradient

The background fill that is combined with the pattern

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

### intensity: int

Intensity of the pattern when it is shown above the filled background; 0-100

#### is\_inverted: bool | None

*Optional*. True, if the background fill must be applied only to the pattern itself. All other pixels are black in this case. For dark themes only

#### is\_moving: bool | None

Optional. True, if the background moves slightly when the device is tilted

#### BackgroundTypeWallpaper

```
class aiogram.types.background_type_wallpaper.BackgroundTypeWallpaper(*, type:
                                                                                     Literal['wallpaper'] =
                                                                                     'wallpaper', document:
                                                                                     Document,
                                                                                     dark_theme_dimming:
                                                                                     int, is blurred: bool |
                                                                                     None = None, is moving:
                                                                                     bool \mid None = None,
                                                                                     **extra_data: Any)
     The background is a wallpaper in the JPEG format.
     Source: https://core.telegram.org/bots/api#backgroundtypewallpaper
     type: Literal['wallpaper']
          Type of the background, always 'wallpaper'
     document: Document
          Document with the wallpaper
     dark_theme_dimming: int
          Dimming of the background in dark themes, as a percentage; 0-100
     model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
          A dictionary of computed field names and their corresponding ComputedFieldInfo objects.
     model_post_init(context: Any, /) \rightarrow None
          We need to both initialize private attributes and call the user-defined model_post_init method.
     is_blurred: bool | None
          Optional. True, if the wallpaper is downscaled to fit in a 450x450 square and then box-blurred with radius
          12
     is_moving: bool | None
          Optional. True, if the background moves slightly when the device is tilted
Birthdate
class aiogram.types.birthdate.Birthdate(*, day: int, month: int, year: int | None = None, **extra_data:
                                                Any)
     Describes the birthdate of a user.
     Source: https://core.telegram.org/bots/api#birthdate
     day: int
          Day of the user's birth; 1-31
     month: int
          Month of the user's birth; 1-12
     model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
          A dictionary of computed field names and their corresponding ComputedFieldInfo objects.
     model_post_init(context: Any, /) \rightarrow None
          We need to both initialize private attributes and call the user-defined model_post_init method.
```

#### year: int | None

Optional. Year of the user's birth

#### **BotCommand**

class aiogram.types.bot\_command.BotCommand(\*, command: str, description: str, \*\*extra\_data: Any)

This object represents a bot command.

Source: https://core.telegram.org/bots/api#botcommand

command: str

Text of the command; 1-32 characters. Can contain only lowercase English letters, digits and underscores.

model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

 $model_post_init(context: Any, /) \rightarrow None$ 

We need to both initialize private attributes and call the user-defined model post init method.

description: str

Description of the command; 1-256 characters.

#### **BotCommandScope**

class aiogram.types.bot\_command\_scope.BotCommandScope(\*\*extra\_data: Any)

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

- $\bullet \ aiogram.types.bot\_command\_scope\_default.BotCommandScopeDefault\\$
- aiogram.types.bot\_command\_scope\_all\_private\_chats.BotCommandScopeAllPrivateChats
- aiogram.types.bot\_command\_scope\_all\_group\_chats.BotCommandScopeAllGroupChats
- aiogram.types.bot\_command\_scope\_all\_chat\_administrators. BotCommandScopeAllChatAdministrators
- aiogram.types.bot\_command\_scope\_chat.BotCommandScopeChat
- aiogram.types.bot\_command\_scope\_chat\_administrators.BotCommandScopeChatAdministrators
- aiogram.types.bot\_command\_scope\_chat\_member.BotCommandScopeChatMember

Source: https://core.telegram.org/bots/api#botcommandscope

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### **BotCommandScopeAllChatAdministrators**

```
\textbf{class} \ \texttt{aiogram.types.bot\_command\_scope\_all\_chat\_administrators.} \\ \textbf{BotCommandScopeAllChatAdministrators} (*, \texttt{administrators}) (*, \texttt{administrators})
```

type: Literal[l

Bot-Command Scope \*\*ex-

Any)

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

Source: https://core.telegram.org/bots/api#botcommandscopeallchatadministrators

type: Literal[BotCommandScopeType.ALL\_CHAT\_ADMINISTRATORS]

Scope type, must be *all\_chat\_administrators* 

model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

 $model_post_init(context: Any, /) \rightarrow None$ 

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### **BotCommandScopeAllGroupChats**

```
class aiogram.types.bot_command_scope_all_group_chats.BotCommandScopeAllGroupChats(*, type:
```

Literal[BotCommandScopeTy
= BotCommandScopeType.ALL\_GROUP\_
\*\*ex-

tra\_data: Any)

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

Source: https://core.telegram.org/bots/api#botcommandscopeallgroupchats

type: Literal[BotCommandScopeType.ALL\_GROUP\_CHATS]

Scope type, must be all\_group\_chats

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

 $model_post_init(context: Any, /) \rightarrow None$ 

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### **BotCommandScopeAllPrivateChats**

```
{\bf class} \ {\bf aiogram.types.bot\_command\_scope\_all\_private\_chats. \\ {\bf BotCommandScopeAllPrivateChats}(*,
```

type: Lit-

eral[BotCommandSc

=

Bot-Com-

mand-

ScopeType.ALL\_PRI

\*\*ex-

tra\_data:
Any)

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

Source: https://core.telegram.org/bots/api#botcommandscopeallprivatechats

type: Literal[BotCommandScopeType.ALL\_PRIVATE\_CHATS]

Scope type, must be all\_private\_chats

model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

 $model_post_init(context: Any, /) \rightarrow None$ 

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### **BotCommandScopeChat**

class aiogram.types.bot\_command\_scope\_chat.BotCommandScopeChat(\*, type: Lit-

eral[BotCommandScopeType.CHAT] = BotCommandScopeType.CHAT, chat\_id: int | str, \*\*extra\_data: Any)

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

Source: https://core.telegram.org/bots/api#botcommandscopechat

type: Literal[BotCommandScopeType.CHAT]

Scope type, must be chat

model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

 $model_post_init(context: Any, /) \rightarrow None$ 

We need to both initialize private attributes and call the user-defined model\_post\_init method.

chat\_id: int | str

Unique identifier for the target chat or username of the target supergroup (in the format @supergroupusername)

#### **BotCommandScopeChatAdministrators**

```
class aiogram.types.bot_command_scope_chat_administrators.BotCommandScopeChatAdministrators(*,
```

```
type:
Lit-
eral[BotComm
=
Bot-
Com-
mand-
ScopeType.CH
chat_id:
int
|
str,
**ex-
tra_data:
Any)
```

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

Source: https://core.telegram.org/bots/api#botcommandscopechatadministrators

### type: Literal[BotCommandScopeType.CHAT\_ADMINISTRATORS]

Scope type, must be chat\_administrators

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
chat_id: int | str
```

Unique identifier for the target chat or username of the target supergroup (in the format @supergroupusername)

### **BotCommandScopeChatMember**

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

Source: https://core.telegram.org/bots/api#botcommandscopechatmember

#### type: Literal[BotCommandScopeType.CHAT\_MEMBER]

Scope type, must be *chat\_member* 

```
chat_id: int | str
```

Unique identifier for the target chat or username of the target supergroup (in the format @supergroupusername)

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
user_id: int
```

Unique identifier of the target user

#### **BotCommandScopeDefault**

```
class aiogram.types.bot_command_scope_default.BotCommandScopeDefault(*, type: Lit-
```

eral[BotCommandScopeType.DEFAULT]

= BotCommand-ScopeType.DEFAULT, \*\*extra\_data: Any)

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

Source: https://core.telegram.org/bots/api#botcommandscopedefault

type: Literal[BotCommandScopeType.DEFAULT]

Scope type, must be default

model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

 $model_post_init(context: Any, /) \rightarrow None$ 

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### **BotDescription**

class aiogram.types.bot\_description.BotDescription(\*, description: str, \*\*extra\_data: Any)

This object represents the bot's description.

Source: https://core.telegram.org/bots/api#botdescription

description: str

The bot's description

model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

 $model_post_init(context: Any, /) \rightarrow None$ 

We need to both initialize private attributes and call the user-defined model\_post\_init method.

# **BotName**

```
class aiogram.types.bot_name.BotName(*, name: str, **extra_data: Any)

This object represents the bot's name.

Source: https://core.telegram.org/bots/api#botname

name: str

The bot's name

model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

model_post_init(context: Any, /) → None

We need to both initialize private attributes and call the user-defined model_post_init method.
```

# **BotShortDescription**

This object represents the bot's short description.

Source: https://core.telegram.org/bots/api#botshortdescription

```
short_description: str

The bot's short description
```

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

## **BusinessConnection**

Describes the connection of the bot with a business account.

Source: https://core.telegram.org/bots/api#businessconnection

```
id: str
```

Unique identifier of the business connection

user: *User* 

Business account user that created the business connection

```
user_chat_id: int
```

Identifier of a private chat with the user who created the business connection. This number may have more than 32 significant bits and some programming languages may have difficulty/silent defects in interpreting it. But it has at most 52 significant bits, so a 64-bit integer or double-precision float type are safe for storing this identifier.

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

date: DateTime

Date the connection was established in Unix time

can\_reply: bool

True, if the bot can act on behalf of the business account in chats that were active in the last 24 hours

is\_enabled: bool

True, if the connection is active

#### **BusinessIntro**

Contains information about the start page settings of a Telegram Business account.

Source: https://core.telegram.org/bots/api#businessintro

title: str | None

Optional. Title text of the business intro

message: str | None

Optional. Message text of the business intro

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

sticker: Sticker | None

Optional. Sticker of the business intro

#### **BusinessLocation**

Contains information about the location of a Telegram Business account.

Source: https://core.telegram.org/bots/api#businesslocation

address: str

Address of the business

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

 $model_post_init(context: Any, /) \rightarrow None$ 

We need to both initialize private attributes and call the user-defined model\_post\_init method.

**location:** Location | None Optional. Location of the business

# **BusinessMessagesDeleted**

class aiogram.types.business\_messages\_deleted.BusinessMessagesDeleted(\*,

business\_connection\_id: str, chat: Chat, message\_ids: List[int], \*\*extra\_data: Any)

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

Source: https://core.telegram.org/bots/api#businessmessagesdeleted

business\_connection\_id: str

Unique identifier of the business connection

chat: Chat

Information about a chat in the business account. The bot may not have access to the chat or the corresponding user.

model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

 $model_post_init(context: Any, /) \rightarrow None$ 

We need to both initialize private attributes and call the user-defined model\_post\_init method.

message\_ids: List[int]

The list of identifiers of deleted messages in the chat of the business account

## **BusinessOpeningHours**

Describes the opening hours of a business.

Source: https://core.telegram.org/bots/api#businessopeninghours

time\_zone\_name: str

Unique name of the time zone for which the opening hours are defined

model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

 $model_post_init(context: Any, /) \rightarrow None$ 

We need to both initialize private attributes and call the user-defined model\_post\_init method.

opening\_hours: List[BusinessOpeningHoursInterval]

List of time intervals describing business opening hours

# **BusinessOpeningHoursInterval**

class aiogram.types.business\_opening\_hours\_interval.BusinessOpeningHoursInterval(\*, open-

ing minute: int, closing\_minute: int, \*\*extra data: Any)

Describes an interval of time during which a business is open.

Source: https://core.telegram.org/bots/api#businessopeninghoursinterval

# opening\_minute: int

The minute's sequence number in a week, starting on Monday, marking the start of the time interval during which the business is open; 0 - 7 \* 24 \* 60

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

# closing\_minute: int

The minute's sequence number in a week, starting on Monday, marking the end of the time interval during which the business is open; 0 - 8 \* 24 \* 60

# CallbackQuery

```
class aiogram.types.callback_query.CallbackQuery(*, id: str, from_user: User, chat_instance: str,
                                                           message: Message | InaccessibleMessage | None =
                                                           None, inline\_message\_id: str \mid None = None, data:
                                                            str | None = None, game_short_name: str | None =
                                                           None, **extra data: Any)
```

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.

NOTE: After the user presses a callback button, Telegram clients will display a progress bar until you call aiogram.methods.answer\_callback\_query.AnswerCallbackQuery. It is, therefore, necessary to react by calling aiogram.methods.answer\_callback\_query.AnswerCallbackQuery even if no notification to the user is needed (e.g., without specifying any of the optional parameters).

Source: https://core.telegram.org/bots/api#callbackquery

# id: str

Unique identifier for this query

from\_user:

User

Sender

## chat\_instance: str

Global identifier, uniquely corresponding to the chat to which the message with the callback button was sent. Useful for high scores in aiogram.methods.games.Games.

### message: Message | InaccessibleMessage | None

Optional. Message sent by the bot with the callback button that originated the query

# model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

## inline\_message\_id: str | None

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

# data: str | None

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

# game\_short\_name: str | None

Optional. Short name of a Game to be returned, serves as the unique identifier for the game

```
answer(text: str \mid None = None, show\_alert: bool \mid None = None, url: <math>str \mid None = None, cache\_time: int \mid None = None, **kwargs: Any) \rightarrow AnswerCallbackQuery
```

Shortcut for method <code>aiogram.methods.answer\_callback\_query.AnswerCallbackQuery</code> will automatically fill method attributes:

• callback\_query\_id

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. On success, True is returned.

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** Text of the notification. If not specified, nothing will be shown to the user, 0-200 characters
- **show\_alert** 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 URL that will be opened by the user's client. If you have created a aiogram.types. game. Game and accepted the conditions via @BotFather, specify the URL that opens your game note that this will only work if the query comes from a https://core.telegram.org/bots/api#inlinekeyboardbutton callback\_game button.
- cache\_time The maximum amount of time in seconds that the result of the callback query may be cached client-side. Telegram apps will support caching starting in version 3.14. Defaults to 0.

#### Returns

```
\begin{array}{ll} instance & of & method \\ & aiogram.methods.answer\_callback\_query. \\ \\ AnswerCallbackQuery \end{array}
```

# Chat

**class** aiogram.types.chat.**Chat**(\*, id: int, type: str, title: str | None = None, username: str | None = None, first\_name: str | None = None, last\_name: str | None = None, is\_forum: bool | *None* = *None*, accent\_color\_id: int | *None* = *None*, active\_usernames: *List[str]* | *None* = *None*, *available\_reactions: List[*ReactionTypeEmoji | ReactionTypeCustomEmoji | ReactionTypePaid] | None = None, background custom emoji id: str | None = None, bio: str | None = None, birthdate: Birthdate | None = None, business intro: BusinessIntro | None = *None, business location:* BusinessLocation | None = None, business\_opening\_hours: BusinessOpeningHours | None = None, can\_set\_sticker\_set: bool | None = None, custom\_emoji\_sticker\_set\_name:  $str \mid None = None, description: str \mid None = None,$ emoji status custom emoji id: str | None = None, emoji status expiration date: datetime serializer, return type=int,  $when\_used=unless - none)$  | None = None,  $has\_aggressive\_anti\_spam\_enabled: bool | None = None,$ has\_hidden\_members: bool | None = None, has\_private\_forwards: bool |  $None = None, has\_protected\_content: bool | None = None,$ has\_restricted\_voice\_and\_video\_messages: bool | None = None, has visible history: bool | None = None, invite link: str | None = None, join\_by\_request: bool | None = None, join\_to\_send\_messages: bool | None = None, linked\_chat\_id: int | None = None, location: ChatLocation | None = *None*, *message\_auto\_delete\_time*: *int* | *None* = *None*, *permissions*: ChatPermissions | None = None, personal chat: Chat | None = None, photo: ChatPhoto | None = None, pinned message: Message | None = None, profile accent color id: int | None = None, profile\_background\_custom\_emoji\_id: str | None = None, slow\_mode\_delay:  $int \mid None = None, sticker\_set\_name: str \mid None = None,$ unrestrict\_boost\_count: int | None = None, \*\*extra\_data: Any)

This object represents a chat.

Source: https://core.telegram.org/bots/api#chat

#### id: int

Unique identifier for this chat. This number may have more than 32 significant bits and some programming languages may have difficulty/silent defects in interpreting it. But it has at most 52 significant bits, so a signed 64-bit integer or double-precision float type are safe for storing this identifier.

# type: str

Type of the chat, can be either 'private', 'group', 'supergroup' or 'channel'

#### title: str | None

Optional. Title, for supergroups, channels and group chats

## username: str | None

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

# first\_name: str | None

Optional. First name of the other party in a private chat

# last\_name: str | None

Optional. Last name of the other party in a private chat

# is\_forum: bool | None

Optional. True, if the supergroup chat is a forum (has topics enabled)

# accent\_color\_id: int | None

Optional. Identifier of the accent color for the chat name and backgrounds of the chat photo, reply header, and link preview. See accent colors for more details. Returned only in aiogram.methods.get\_chat. GetChat. Always returned in aiogram.methods.get\_chat.GetChat.

Deprecated since version API:7.3: https://core.telegram.org/bots/api-changelog#may-6-2024

# active\_usernames: List[str] | None

*Optional*. If non-empty, the list of all active chat usernames; for private chats, supergroups and channels. Returned only in aiogram.methods.get\_chat.GetChat.

Deprecated since version API:7.3: https://core.telegram.org/bots/api-changelog#may-6-2024

# available\_reactions: List[ReactionTypeEmoji | ReactionTypeCustomEmoji | ReactionTypePaid] | None

*Optional*. List of available reactions allowed in the chat. If omitted, then all emoji reactions are allowed. Returned only in aiogram.methods.get\_chat.GetChat.

Deprecated since version API:7.3: https://core.telegram.org/bots/api-changelog#may-6-2024

# background\_custom\_emoji\_id: str | None

*Optional*. Custom emoji identifier of emoji chosen by the chat for the reply header and link preview background. Returned only in aiogram.methods.get\_chat.GetChat.

Deprecated since version API:7.3: https://core.telegram.org/bots/api-changelog#may-6-2024

# bio: str | None

Optional. Bio of the other party in a private chat. Returned only in aiogram.methods.get\_chat. GetChat.

Deprecated since version API:7.3: https://core.telegram.org/bots/api-changelog#may-6-2024

## birthdate: Birthdate | None

*Optional*. For private chats, the date of birth of the user. Returned only in aiogram.methods.get\_chat. GetChat.

Deprecated since version API:7.3: https://core.telegram.org/bots/api-changelog#may-6-2024

# business\_intro: BusinessIntro | None

*Optional*. For private chats with business accounts, the intro of the business. Returned only in *aiogram*. methods.get\_chat.GetChat.

Deprecated since version API:7.3: https://core.telegram.org/bots/api-changelog#may-6-2024

# business\_location: BusinessLocation | None

*Optional*. For private chats with business accounts, the location of the business. Returned only in *aiogram*. methods.get\_chat.GetChat.

Deprecated since version API:7.3: https://core.telegram.org/bots/api-changelog#may-6-2024

# business\_opening\_hours: BusinessOpeningHours | None

*Optional*. For private chats with business accounts, the opening hours of the business. Returned only in aiogram.methods.get\_chat.GetChat.

Deprecated since version API:7.3: https://core.telegram.org/bots/api-changelog#may-6-2024

# can\_set\_sticker\_set: bool | None

*Optional.* True, if the bot can change the group sticker set. Returned only in aiogram.methods. get\_chat.GetChat.

Deprecated since version API:7.3: https://core.telegram.org/bots/api-changelog#may-6-2024

### custom\_emoji\_sticker\_set\_name: str | None

*Optional*. For supergroups, the name of the group's custom emoji sticker set. Custom emoji from this set can be used by all users and bots in the group. Returned only in aiogram.methods.get\_chat.GetChat.

Deprecated since version API:7.3: https://core.telegram.org/bots/api-changelog#may-6-2024

# description: str | None

*Optional*. Description, for groups, supergroups and channel chats. Returned only in aiogram.methods. get\_chat.GetChat.

Deprecated since version API:7.3: https://core.telegram.org/bots/api-changelog#may-6-2024

# emoji\_status\_custom\_emoji\_id: str | None

*Optional*. Custom emoji identifier of the emoji status of the chat or the other party in a private chat. Returned only in aiogram.methods.get\_chat.GetChat.

Deprecated since version API:7.3: https://core.telegram.org/bots/api-changelog#may-6-2024

# emoji\_status\_expiration\_date: DateTime | None

*Optional*. Expiration date of the emoji status of the chat or the other party in a private chat, in Unix time, if any. Returned only in aiogram.methods.get\_chat.GetChat.

Deprecated since version API:7.3: https://core.telegram.org/bots/api-changelog#may-6-2024

# has\_aggressive\_anti\_spam\_enabled: bool | None

*Optional.* True, if aggressive anti-spam checks are enabled in the supergroup. The field is only available to chat administrators. Returned only in aiogram.methods.get\_chat.GetChat.

Deprecated since version API:7.3: https://core.telegram.org/bots/api-changelog#may-6-2024

### has\_hidden\_members: bool | None

*Optional*. True, if non-administrators can only get the list of bots and administrators in the chat. Returned only in <code>aiogram.methods.get\_chat.GetChat</code>.

Deprecated since version API:7.3: https://core.telegram.org/bots/api-changelog#may-6-2024

# has\_private\_forwards: bool | None

Optional. True, if privacy settings of the other party in the private chat allows to use tg://user?id=<user\_id> links only in chats with the user. Returned only in aiogram.methods.get\_chat.

GetChat.

Deprecated since version API:7.3: https://core.telegram.org/bots/api-changelog#may-6-2024

# has\_protected\_content: bool | None

*Optional*. True, if messages from the chat can't be forwarded to other chats. Returned only in *aiogram*. methods.get\_chat.GetChat.

Deprecated since version API:7.3: https://core.telegram.org/bots/api-changelog#may-6-2024

# has\_restricted\_voice\_and\_video\_messages: bool | None

*Optional*. True, if the privacy settings of the other party restrict sending voice and video note messages in the private chat. Returned only in aiogram.methods.get\_chat.GetChat.

Deprecated since version API:7.3: https://core.telegram.org/bots/api-changelog#may-6-2024

# has\_visible\_history: bool | None

*Optional*. True, if new chat members will have access to old messages; available only to chat administrators. Returned only in aiogram.methods.get\_chat.GetChat.

Deprecated since version API:7.3: https://core.telegram.org/bots/api-changelog#may-6-2024

# invite\_link: str | None

*Optional*. Primary invite link, for groups, supergroups and channel chats. Returned only in *aiogram.* methods.get\_chat.GetChat.

Deprecated since version API:7.3: https://core.telegram.org/bots/api-changelog#may-6-2024

# join\_by\_request: bool | None

*Optional*. True, if all users directly joining the supergroup need to be approved by supergroup administrators. Returned only in aiogram.methods.get\_chat.GetChat.

Deprecated since version API:7.3: https://core.telegram.org/bots/api-changelog#may-6-2024

# join\_to\_send\_messages: bool | None

*Optional*. True, if users need to join the supergroup before they can send messages. Returned only in aiogram.methods.get\_chat.GetChat.

Deprecated since version API:7.3: https://core.telegram.org/bots/api-changelog#may-6-2024

# linked\_chat\_id: int | None

Optional. Unique identifier for the linked chat, i.e. the discussion group identifier for a channel and vice versa; for supergroups and channel chats. This identifier may be greater than 32 bits and some programming languages may have difficulty/silent defects in interpreting it. But it is smaller than 52 bits, so a signed 64 bit integer or double-precision float type are safe for storing this identifier. Returned only in aiogram. methods.get\_chat.GetChat.

Deprecated since version API:7.3: https://core.telegram.org/bots/api-changelog#may-6-2024

#### location: ChatLocation | None

*Optional.* For supergroups, the location to which the supergroup is connected. Returned only in *aiogram.* methods.get\_chat.GetChat.

Deprecated since version API:7.3: https://core.telegram.org/bots/api-changelog#may-6-2024

# message\_auto\_delete\_time: int | None

*Optional*. The time after which all messages sent to the chat will be automatically deleted; in seconds. Returned only in aiogram.methods.get\_chat.GetChat.

Deprecated since version API:7.3: https://core.telegram.org/bots/api-changelog#may-6-2024

# permissions: ChatPermissions | None

Optional. Default chat member permissions, for groups and supergroups. Returned only in aiogram.  $methods.get\_chat.GetChat.$ 

Deprecated since version API:7.3: https://core.telegram.org/bots/api-changelog#may-6-2024

## personal\_chat: Chat | None

*Optional.* For private chats, the personal channel of the user. Returned only in aiogram.methods. get\_chat.GetChat.

Deprecated since version API:7.3: https://core.telegram.org/bots/api-changelog#may-6-2024

# photo: ChatPhoto | None

Optional. Chat photo. Returned only in aiogram.methods.get\_chat.GetChat.

Deprecated since version API:7.3: https://core.telegram.org/bots/api-changelog#may-6-2024

# model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

## $model_post_init(context: Any, /) \rightarrow None$

We need to both initialize private attributes and call the user-defined model\_post\_init method.

# pinned\_message: Message | None

*Optional*. The most recent pinned message (by sending date). Returned only in *aiogram.methods*.  $get\_chat.GetChat$ .

Deprecated since version API:7.3: https://core.telegram.org/bots/api-changelog#may-6-2024

## profile\_accent\_color\_id: int | None

*Optional*. Identifier of the accent color for the chat's profile background. See profile accent colors for more details. Returned only in aiogram.methods.get\_chat.GetChat.

Deprecated since version API:7.3: https://core.telegram.org/bots/api-changelog#may-6-2024

# profile\_background\_custom\_emoji\_id: str | None

*Optional*. Custom emoji identifier of the emoji chosen by the chat for its profile background. Returned only in aiogram.methods.get\_chat.GetChat.

Deprecated since version API:7.3: https://core.telegram.org/bots/api-changelog#may-6-2024

# slow\_mode\_delay: int | None

*Optional*. For supergroups, the minimum allowed delay between consecutive messages sent by each unprivileged user; in seconds. Returned only in aiogram.methods.get\_chat.GetChat.

Deprecated since version API:7.3: https://core.telegram.org/bots/api-changelog#may-6-2024

# sticker\_set\_name: str | None

*Optional.* For supergroups, name of group sticker set. Returned only in aiogram.methods.get\_chat. GetChat.

Deprecated since version API:7.3: https://core.telegram.org/bots/api-changelog#may-6-2024

# unrestrict\_boost\_count: int | None

*Optional.* For supergroups, the minimum number of boosts that a non-administrator user needs to add in order to ignore slow mode and chat permissions. Returned only in aiogram.methods.get\_chat. GetChat.

Deprecated since version API:7.3: https://core.telegram.org/bots/api-changelog#may-6-2024

# property shifted\_id: int

Returns shifted chat ID (positive and without "-100" prefix). Mostly used for private links like t.me/c/chat\_id/message\_id

Currently supergroup/channel IDs have 10-digit ID after "-100" prefix removed. However, these IDs might become 11-digit in future. So, first we remove "-100" prefix and count remaining number length. Then we multiple -1 \* 10 (number\_length + 2) Finally, self.id is substracted from that number

# property full\_name: str

Get full name of the Chat.

For private chat it is first\_name + last\_name. For other chat types it is title.

# $ban\_sender\_chat(sender\_chat\_id: int, **kwargs: Any) \rightarrow BanChatSenderChat$

Shortcut for method aiogram.methods.ban\_chat\_sender\_chat.BanChatSenderChat will automatically fill method attributes:

chat\_id

Use this method to ban a channel chat in a supergroup or a channel. Until the chat is unbanned, the owner of the banned chat won't be able to send messages on behalf of **any of their channels**. The bot must be an administrator in the supergroup or channel for this to work and must have the appropriate administrator rights. Returns True on success.

Source: https://core.telegram.org/bots/api#banchatsenderchat

#### **Parameters**

**sender\_chat\_id** – Unique identifier of the target sender chat

#### Returns

instance of method aiogram.methods.ban\_chat\_sender\_chat.BanChatSenderChat

```
unban_sender_chat(sender_chat_id: int, **kwargs: Any) → UnbanChatSenderChat
```

Shortcut for method <code>aiogram.methods.unban\_chat\_sender\_chat.UnbanChatSenderChat</code> will automatically fill method attributes:

• chat\_id

Use this method to unban a previously banned channel chat in a supergroup or channel. The bot must be an administrator for this to work and must have the appropriate administrator rights. Returns True on success.

Source: https://core.telegram.org/bots/api#unbanchatsenderchat

#### **Parameters**

**sender\_chat\_id** – Unique identifier of the target sender chat

#### Returns

instance of method aiogram.methods.unban\_chat\_sender\_chat.
UnbanChatSenderChat

# $get_administrators(**kwargs: Any) \rightarrow GetChatAdministrators$

Shortcut for method  $aiogram.methods.get\_chat\_administrators.GetChatAdministrators$  will automatically fill method attributes:

• chat\_id

Use this method to get a list of administrators in a chat, which aren't bots. Returns an Array of aiogram. types.chat\_member.ChatMember objects.

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

# Returns

instance of method aiogram.methods.get\_chat\_administrators.
GetChatAdministrators

```
delete_message(message\_id: int, **kwargs: Any) \rightarrow DeleteMessage
```

Shortcut for method aiogram.methods.delete\_message.DeleteMessage will automatically fill method attributes:

• chat\_id

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.
- Service messages about a supergroup, channel, or forum topic creation can't be deleted.
- A dice message in a private chat can only be deleted if it was sent more than 24 hours ago.
- Bots can delete outgoing messages in private chats, groups, and supergroups.
- Bots can delete incoming messages in private chats.
- Bots granted can\_post\_messages permissions can delete outgoing messages in channels.

- If the bot is an administrator of a group, it can delete any message there.
- If the bot has can\_delete\_messages permission in a supergroup or a channel, it can delete any message there.

Returns True on success.

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

#### **Parameters**

**message\_id** – Identifier of the message to delete

#### Returns

instance of method aiogram.methods.delete\_message.DeleteMessage

```
revoke\_invite\_link(invite\_link: str, **kwargs: Any) \rightarrow RevokeChatInviteLink
```

Shortcut for method aiogram.methods.revoke\_chat\_invite\_link.RevokeChatInviteLink will automatically fill method attributes:

• chat\_id

Use this method to revoke an invite link created by the bot. If the primary link is revoked, a new link is automatically generated. The bot must be an administrator in the chat for this to work and must have the appropriate administrator rights. Returns the revoked invite link as <code>aiogram.types.chat\_invite\_link</code>. <code>ChatInviteLink</code> object.

Source: https://core.telegram.org/bots/api#revokechatinvitelink

#### Parameters

**invite\_link** – The invite link to revoke

## Returns

 $\begin{array}{ll} \text{instance} & \text{of} & \text{method} & \textit{aiogram.methods.revoke\_chat\_invite\_link.} \\ \textit{RevokeChatInviteLink} & \end{array}$ 

```
edit_invite_link(invite_link: str, name: str | None = None, expire_date: datetime.datetime | datetime.timedelta | int | None = None, member_limit: int | None = None, creates_join_request: bool | None = None, **kwargs: Any) → EditChatInviteLink
```

Shortcut for method <code>aiogram.methods.edit\_chat\_invite\_link.EditChatInviteLink</code> will automatically fill method attributes:

• chat\_id

Use this method to edit a non-primary invite link created by the bot. The bot must be an administrator in the chat for this to work and must have the appropriate administrator rights. Returns the edited invite link as a <code>aiogram.types.chat\_invite\_link.ChatInviteLink</code> object.

Source: https://core.telegram.org/bots/api#editchatinvitelink

## **Parameters**

- invite\_link The invite link to edit
- name Invite link name; 0-32 characters
- expire\_date Point in time (Unix timestamp) when the link will expire
- member\_limit The maximum number of users that can be members of the chat simultaneously after joining the chat via this invite link; 1-99999
- **creates\_join\_request** True, if users joining the chat via the link need to be approved by chat administrators. If True, *member\_limit* can't be specified

## Returns

```
instance of method aiogram.methods.edit_chat_invite_link.
EditChatInviteLink
```

**create\_invite\_link**(name:  $str \mid None = None$ ,  $expire\_date$ :  $datetime.datetime \mid datetime.timedelta \mid int \mid None = None$ ,  $member\_limit$ :  $int \mid None = None$ ,  $creates\_join\_request$ :  $bool \mid None = None$ , \*\*kwargs: Any)  $\rightarrow CreateChatInviteLink$ 

Shortcut for method aiogram.methods.create\_chat\_invite\_link.CreateChatInviteLink will automatically fill method attributes:

chat\_id

Use this method to create an additional invite link for a chat. The bot must be an administrator in the chat for this to work and must have the appropriate administrator rights. The link can be revoked using the method <code>aiogram.methods.revoke\_chat\_invite\_link.RevokeChatInviteLink</code>. Returns the new invite link as <code>aiogram.types.chat\_invite\_link.ChatInviteLink</code> object.

Source: https://core.telegram.org/bots/api#createchatinvitelink

## **Parameters**

- name Invite link name; 0-32 characters
- **expire\_date** Point in time (Unix timestamp) when the link will expire
- member\_limit The maximum number of users that can be members of the chat simultaneously after joining the chat via this invite link; 1-99999
- **creates\_join\_request** True, if users joining the chat via the link need to be approved by chat administrators. If True, *member\_limit* can't be specified

## Returns

```
instance of method aiogram.methods.create_chat_invite_link.
CreateChatInviteLink
```

```
export_invite_link(**kwargs: Any) \rightarrow ExportChatInviteLink
```

Shortcut for method  $aiogram.methods.export\_chat\_invite\_link.ExportChatInviteLink$  will automatically fill method attributes:

• chat\_id

Use this method to generate a new primary invite link for a chat; any previously generated primary link is revoked. The bot must be an administrator in the chat for this to work and must have the appropriate administrator rights. Returns the new invite link as *String* on success.

Note: Each administrator in a chat generates their own invite links. Bots can't use invite links generated by other administrators. If you want your bot to work with invite links, it will need to generate its own link using <code>aiogram.methods.export\_chat\_invite\_link</code>. <code>ExportChatInviteLink</code> or by calling the <code>aiogram.methods.get\_chat.GetChat</code> method. If your bot needs to generate a new primary invite link replacing its previous one, use <code>aiogram.methods.export\_chat\_invite\_link.ExportChatInviteLink</code> again.

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

## Returns

```
instance of method aiogram.methods.export_chat_invite_link.
ExportChatInviteLink
```

**do**(action: str, business\_connection\_id: str | None = None, message\_thread\_id: int | None = None, \*\*kwargs: Any)  $\rightarrow$  SendChatAction

Shortcut for method <code>aiogram.methods.send\_chat\_action.SendChatAction</code> will automatically fill method attributes:

#### • chat\_id

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

Example: The ImageBot needs some time to process a request and upload the image. Instead of sending a text message along the lines of 'Retrieving image, please wait...', the bot may use aiogram.methods.send\_chat\_action.SendChatAction with action = upload\_photo. The user will see a 'sending photo' status for the bot.

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 Type of action to broadcast. Choose one, depending on what the user is about to receive: typing for text messages, upload\_photo for photos, record\_video or upload\_video for videos, record\_voice or upload\_voice for voice notes, upload\_document for general files, choose\_sticker for stickers, find\_location for location data, record\_video\_note or upload\_video\_note for video notes.
- business\_connection\_id Unique identifier of the business connection on behalf of which the action will be sent
- message\_thread\_id Unique identifier for the target message thread; for supergroups only

#### Returns

instance of method aiogram.methods.send\_chat\_action.SendChatAction

# $delete\_sticker\_set(**kwargs: Any) \rightarrow DeleteChatStickerSet$

Shortcut for method aiogram.methods.delete\_chat\_sticker\_set.DeleteChatStickerSet will automatically fill method attributes:

• chat\_id

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 administrator rights. Use the field <code>can\_set\_sticker\_set</code> optionally returned in <code>aiogram.methods.get\_chat.GetChat</code> requests to check if the bot can use this method. Returns True on success.

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

#### Returns

instance of method aiogram.methods.delete\_chat\_sticker\_set.
DeleteChatStickerSet

 $set\_sticker\_set(sticker\_set\_name: str, **kwargs: Any) \rightarrow SetChatStickerSet$ 

Shortcut for method aiogram.methods.set\_chat\_sticker\_set.SetChatStickerSet will automatically fill method attributes:

• chat\_id

Use this method to set a new group sticker set for a supergroup. The bot must be an administrator in the chat for this to work and must have the appropriate administrator rights. Use the field <code>can\_set\_sticker\_set</code> optionally returned in <code>aiogram.methods.get\_chat.GetChat</code> requests to check if the bot can use this method. Returns True on success.

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

#### **Parameters**

**sticker\_set\_name** – Name of the sticker set to be set as the group sticker set

#### Returns

instance of method aiogram.methods.set\_chat\_sticker\_set.SetChatStickerSet

```
get_member(user\ id:\ int,\ **kwargs:\ Any) \rightarrow GetChatMember
```

Shortcut for method  $aiogram.methods.get\_chat\_member.GetChatMember$  will automatically fill method attributes:

• chat\_id

Use this method to get information about a member of a chat. The method is only guaranteed to work for other users if the bot is an administrator in the chat. Returns a aiogram.types.chat\_member. ChatMember object on success.

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

#### **Parameters**

**user\_id** – Unique identifier of the target user

#### Returns

instance of method aiogram.methods.get\_chat\_member.GetChatMember

```
get_member_count(**kwargs: Any) \rightarrow GetChatMemberCount
```

Shortcut for method  $aiogram.methods.get\_chat\_member\_count.GetChatMemberCount$  will automatically fill method attributes:

• chat\_id

Use this method to get the number of members in a chat. Returns *Int* on success.

Source: https://core.telegram.org/bots/api#getchatmembercount

# Returns

instance of method aiogram.methods.get\_chat\_member\_count.
GetChatMemberCount

```
leave(**kwargs: Any) \rightarrow LeaveChat
```

Shortcut for method aiogram.methods.leave\_chat.LeaveChat will automatically fill method attributes:

• chat\_id

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

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

## **Returns**

instance of method aiogram.methods.leave\_chat.LeaveChat

```
unpin\_all\_messages(**kwargs: Any) \rightarrow UnpinAllChatMessages
```

 $Shortcut\ for\ method\ aiogram.methods.unpin\_all\_chat\_messages.UnpinAllChatMessages\ will automatically fill\ method\ attributes:$ 

• chat\_id

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' administrator right in a supergroup or 'can\_edit\_messages' administrator right in a channel. Returns True on success.

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

## Returns

instance of method aiogram.methods.unpin\_all\_chat\_messages. UnpinAllChatMessages

**unpin\_message**(business\_connection\_id:  $str \mid None = None, message_id: int \mid None = None, **kwargs: Any) <math>\rightarrow UnpinChatMessage$ 

Shortcut for method  $aiogram.methods.unpin\_chat\_message.UnpinChatMessage$  will automatically fill method attributes:

· chat id

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' administrator right in a supergroup or 'can\_edit\_messages' administrator right in a channel. Returns True on success.

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

## **Parameters**

- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be unpinned
- message\_id Identifier of the message to unpin. Required if business\_connection\_id is specified. If not specified, the most recent pinned message (by sending date) will be unpinned.

#### Returns

instance of method aiogram.methods.unpin\_chat\_message.UnpinChatMessage

 $pin_message(message\_id: int, business\_connection\_id: str | None = None, disable\_notification: bool | None = None, **kwargs: Any) <math>\rightarrow PinChatMessage$ 

Shortcut for method <code>aiogram.methods.pin\_chat\_message.PinChatMessage</code> will automatically fill method attributes:

chat\_id

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' administrator right in a supergroup or 'can\_edit\_messages' administrator right in a channel. Returns True on success.

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

## **Parameters**

- message\_id Identifier of a message to pin
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be pinned
- disable\_notification Pass True if it is not necessary to send a notification to all chat
  members about the new pinned message. Notifications are always disabled in channels and
  private chats.

# Returns

instance of method aiogram.methods.pin\_chat\_message.PinChatMessage

```
set\_administrator\_custom\_title(user\_id: int, custom\_title: str, **kwargs: Any) \rightarrow SetChatAdministratorCustomTitle
```

Shortcut for method aiogram.methods.set\_chat\_administrator\_custom\_title. SetChatAdministratorCustomTitle will automatically fill method attributes:

## • chat\_id

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**

instance of method aiogram.methods.set\_chat\_administrator\_custom\_title.
SetChatAdministratorCustomTitle

**set\_permissions**(permissions: ChatPermissions, use\_independent\_chat\_permissions: bool | None = None, \*\*kwargs: Any)  $\rightarrow$  SetChatPermissions

Shortcut for method <code>aiogram.methods.set\_chat\_permissions.SetChatPermissions</code> will automatically fill method attributes:

• chat\_id

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* administrator rights. Returns True on success.

Source: https://core.telegram.org/bots/api#setchatpermissions

#### **Parameters**

- permissions A JSON-serialized object for new default chat permissions
- use\_independent\_chat\_permissions Pass True if chat permissions are set independently. Otherwise, the can\_send\_other\_messages and can\_add\_web\_page\_previews permissions will imply the can\_send\_messages, can\_send\_audios, can\_send\_documents, can\_send\_photos, can\_send\_videos, can\_send\_video\_notes, and can\_send\_voice\_notes permissions; the can\_send\_polls permission will imply the can\_send\_messages permission.

## Returns

instance of method aiogram.methods.set\_chat\_permissions.SetChatPermissions

Shortcut for method aiogram.methods.promote\_chat\_member.PromoteChatMember will automatically fill method attributes:

• chat\_id

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 administrator rights. Pass False for all boolean parameters to demote a user. Returns True on success.

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

#### **Parameters**

- user\_id Unique identifier of the target user
- is\_anonymous Pass True if the administrator's presence in the chat is hidden
- can\_manage\_chat Pass True if the administrator can access the chat event log, get boost list, see hidden supergroup and channel members, report spam messages and ignore slow mode. Implied by any other administrator privilege.
- can\_delete\_messages Pass True if the administrator can delete messages of other users
- can\_manage\_video\_chats Pass True if the administrator can manage video chats
- **can\_restrict\_members** Pass True if the administrator can restrict, ban or unban chat members, or access supergroup statistics
- **can\_promote\_members** Pass True if the administrator can add new administrators with a subset of their own privileges or demote administrators that they have promoted, directly or indirectly (promoted by administrators that were appointed by him)
- can\_change\_info Pass True if the administrator can change chat title, photo and other settings
- can\_invite\_users Pass True if the administrator can invite new users to the chat
- can\_post\_stories Pass True if the administrator can post stories to the chat
- **can\_edit\_stories** Pass True if the administrator can edit stories posted by other users, post stories to the chat page, pin chat stories, and access the chat's story archive
- can\_delete\_stories Pass True if the administrator can delete stories posted by other users
- **can\_post\_messages** Pass True if the administrator can post messages in the channel, or access channel statistics; for channels only
- can\_edit\_messages Pass True if the administrator can edit messages of other users and can pin messages; for channels only
- **can\_pin\_messages** Pass **True** if the administrator can pin messages; for supergroups only
- can\_manage\_topics Pass True if the user is allowed to create, rename, close, and reopen forum topics; for supergroups only

## **Returns**

instance of method aiogram.methods.promote\_chat\_member.PromoteChatMember

 $\begin{tabular}{ll} \textbf{restrict} (user\_id: int, permissions: ChatPermissions, use\_independent\_chat\_permissions: bool | None = \\ None, until\_date: datetime.datetime | datetime.timedelta | int | None = None, **kwargs: Any) \rightarrow \\ RestrictChatMember \end{tabular}$ 

Shortcut for method  $aiogram.methods.restrict\_chat\_member.RestrictChatMember$  will automatically fill method attributes:

• chat\_id

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 administrator rights. Pass True for all permissions to lift restrictions from a user. Returns True on success.

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

## **Parameters**

- user\_id Unique identifier of the target user
- **permissions** A JSON-serialized object for new user permissions
- use\_independent\_chat\_permissions Pass True if chat permissions are set independently. Otherwise, the can\_send\_other\_messages and can\_add\_web\_page\_previews permissions will imply the can\_send\_messages, can\_send\_audios, can\_send\_documents, can\_send\_photos, can\_send\_videos, can\_send\_video\_notes, and can\_send\_voice\_notes permissions; the can\_send\_polls permission will imply the can\_send\_messages permission.
- until\_date Date when restrictions will be lifted for the user; Unix time. If user is restricted for more than 366 days or less than 30 seconds from the current time, they are considered to be restricted forever

#### Returns

instance of method aiogram.methods.restrict\_chat\_member.RestrictChatMember

**unban**(user\_id: int, only\_if\_banned: bool | None = None, \*\*kwargs: Any) → UnbanChatMember

Shortcut for method aiogram.methods.unban\_chat\_member.UnbanChatMember will automatically fill method attributes:

· chat id

Use this method to unban a previously banned 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 Unique identifier of the target user
- only\_if\_banned Do nothing if the user is not banned

## Returns

 $instance\ of\ method\ aiogram.methods.unban\_chat\_member.UnbanChatMember$ 

 $ban(user\_id: int, until\_date: datetime.datetime \mid datetime.timedelta \mid int \mid None = None, revoke\_messages: bool \mid None = None, **kwargs: Any) \rightarrow BanChatMember$ 

Shortcut for method aiogram.methods.ban\_chat\_member.BanChatMember will automatically fill method attributes:

• chat\_id

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

Source: https://core.telegram.org/bots/api#banchatmember

## **Parameters**

• **user\_id** – Unique identifier of the target user

- until\_date Date when the user will be unbanned; Unix time. If user is banned for more than 366 days or less than 30 seconds from the current time they are considered to be banned forever. Applied for supergroups and channels only.
- **revoke\_messages** Pass True to delete all messages from the chat for the user that is being removed. If False, the user will be able to see messages in the group that were sent before the user was removed. Always True for supergroups and channels.

#### Returns

instance of method aiogram.methods.ban\_chat\_member.BanChatMember

 $set\_description(description: str \mid None = None, **kwargs: Any) \rightarrow SetChatDescription$ 

Shortcut for method aiogram.methods.set\_chat\_description.SetChatDescription will automatically fill method attributes:

• chat\_id

Use this method to change the description of a group, a supergroup or a channel. The bot must be an administrator in the chat for this to work and must have the appropriate administrator rights. Returns True on success.

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

#### **Parameters**

**description** – New chat description, 0-255 characters

#### Returns

 $instance\ of\ method\ aiogram.methods.set\_chat\_description. SetChatDescription$ 

 $set\_title(title: str, **kwargs: Any) \rightarrow SetChatTitle$ 

Shortcut for method  $aiogram.methods.set\_chat\_title.SetChatTitle$  will automatically fill method attributes:

• chat\_id

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 administrator rights. Returns True on success.

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

# **Parameters**

**title** – New chat title, 1-128 characters

#### Returns

instance of method aiogram.methods.set\_chat\_title.SetChatTitle

 $delete\_photo(**kwargs: Any) \rightarrow DeleteChatPhoto$ 

Shortcut for method <code>aiogram.methods.delete\_chat\_photo.DeleteChatPhoto</code> will automatically fill method attributes:

• chat\_id

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 administrator rights. Returns True on success.

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

## Returns

 $instance\ of\ method\ aiogram. \verb|methods.delete_chat_photo.DeleteChatPhoto|\\$ 

 $set\_photo(photo: InputFile, **kwargs: Any) \rightarrow SetChatPhoto$ 

Shortcut for method <code>aiogram.methods.set\_chat\_photo.SetChatPhoto</code> will automatically fill method attributes:

• chat id

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 administrator rights. Returns True on success.

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

#### **Parameters**

photo - New chat photo, uploaded using multipart/form-data

## **Returns**

instance of method aiogram.methods.set\_chat\_photo.SetChatPhoto

 $\begin{tabular}{ll} \bf unpin\_all\_general\_forum\_topic\_messages (**kwargs: Any) \to UnpinAllGeneralForumTopicMessages \\ Shortcut & for method aiogram.methods.unpin\_all\_general\_forum\_topic\_messages. \\ UnpinAllGeneralForumTopicMessages & will automatically fill method attributes: \\ \end{tabular}$ 

chat\_id

Use this method to clear the list of pinned messages in a General forum topic. The bot must be an administrator in the chat for this to work and must have the *can\_pin\_messages* administrator right in the supergroup. Returns True on success.

Source: https://core.telegram.org/bots/api#unpinallgeneralforumtopicmessages

#### Returns

 $instance\ of\ method\ aiogram.methods.unpin\_all\_general\_forum\_topic\_messages.$  UnpinAllGeneralForumTopicMessages

# ChatAdministratorRights

class aiogram.types.chat\_administrator\_rights.ChatAdministratorRights(\*, is\_anonymous: bool,

can\_manage\_chat: bool, can\_delete\_messages:

bool,

can\_manage\_video\_chats:

bool,

can\_restrict\_members:

bool,

can\_promote\_members:

bool, can\_change\_info:

bool, can\_invite\_users:

bool, can\_post\_stories:

bool, can\_edit\_stories:

bool, can\_delete\_stories:

bool,

can\_post\_messages: bool | None = None,

can\_edit\_messages: bool

| None = None,

can\_pin\_messages: bool

| None = None, can\_manage\_topics: bool | None = None, \*\*extra data: Any)

Represents the rights of an administrator in a chat.

Source: https://core.telegram.org/bots/api#chatadministratorrights

## is\_anonymous: bool

True, if the user's presence in the chat is hidden

# can\_manage\_chat: bool

True, if the administrator can access the chat event log, get boost list, see hidden supergroup and channel members, report spam messages and ignore slow mode. Implied by any other administrator privilege.

# can\_delete\_messages: bool

True, if the administrator can delete messages of other users

# can\_manage\_video\_chats: bool

True, if the administrator can manage video chats

# can\_restrict\_members: bool

True, if the administrator can restrict, ban or unban chat members, or access supergroup statistics

## can\_promote\_members: bool

True, if the administrator can add new administrators with a subset of their own privileges or demote administrators that they have promoted, directly or indirectly (promoted by administrators that were appointed by the user)

# can\_change\_info: bool

True, if the user is allowed to change the chat title, photo and other settings

## can\_invite\_users: bool

True, if the user is allowed to invite new users to the chat

# model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

# $model_post_init(context: Any, /) \rightarrow None$

We need to both initialize private attributes and call the user-defined model\_post\_init method.

# can\_post\_stories: bool

True, if the administrator can post stories to the chat

## can\_edit\_stories: bool

True, if the administrator can edit stories posted by other users, post stories to the chat page, pin chat stories, and access the chat's story archive

## can\_delete\_stories: bool

True, if the administrator can delete stories posted by other users

# can\_post\_messages: bool | None

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

# can\_edit\_messages: bool | None

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

# can\_pin\_messages: bool | None

Optional. True, if the user is allowed to pin messages; for groups and supergroups only

# can\_manage\_topics: bool | None

*Optional*. True, if the user is allowed to create, rename, close, and reopen forum topics; for supergroups only

# ChatBackground

```
class aiogram.types.chat_background.ChatBackground(*, type: BackgroundTypeFill |
```

BackgroundTypeWallpaper | BackgroundTypePattern |

BackgroundTypeChatTheme, \*\*extra\_data: Any)

This object represents a chat background.

Source: https://core.telegram.org/bots/api#chatbackground

# type: BackgroundTypeFill | BackgroundTypeWallpaper | BackgroundTypePattern | BackgroundTypeChatTheme

Type of the background

# model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model post init method.

# **ChatBoost**

This object contains information about a chat boost.

Source: https://core.telegram.org/bots/api#chatboost

boost\_id: str

Unique identifier of the boost

add date: DateTime

Point in time (Unix timestamp) when the chat was boosted

model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

 $model_post_init(context: Any, /) \rightarrow None$ 

We need to both initialize private attributes and call the user-defined model\_post\_init method.

expiration\_date: DateTime

Point in time (Unix timestamp) when the boost will automatically expire, unless the booster's Telegram Premium subscription is prolonged

source: ChatBoostSourcePremium | ChatBoostSourceGiftCode | ChatBoostSourceGiveaway
Source of the added boost

# ChatBoostAdded

class aiogram.types.chat\_boost\_added.ChatBoostAdded(\*, boost\_count: int, \*\*extra\_data: Any)

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

Source: https://core.telegram.org/bots/api#chatboostadded

boost\_count: int

Number of boosts added by the user

model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

 $model_post_init(context: Any, /) \rightarrow None$ 

We need to both initialize private attributes and call the user-defined model\_post\_init method.

# ChatBoostRemoved

This object represents a boost removed from a chat.

Source: https://core.telegram.org/bots/api#chatboostremoved

chat: Chat

Chat which was boosted

boost\_id: str

Unique identifier of the boost

model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

 $model_post_init(context: Any, /) \rightarrow None$ 

We need to both initialize private attributes and call the user-defined model\_post\_init method.

remove\_date: DateTime

Point in time (Unix timestamp) when the boost was removed

source: ChatBoostSourcePremium | ChatBoostSourceGiftCode | ChatBoostSourceGiveaway
Source of the removed boost

# **ChatBoostSource**

class aiogram.types.chat\_boost\_source.ChatBoostSource(\*\*extra\_data: Any)

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

- aiogram.types.chat\_boost\_source\_premium.ChatBoostSourcePremium
- aiogram.types.chat\_boost\_source\_gift\_code.ChatBoostSourceGiftCode
- aiogram.types.chat\_boost\_source\_giveaway.ChatBoostSourceGiveaway

Source: https://core.telegram.org/bots/api#chatboostsource

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

## ChatBoostSourceGiftCode

The boost was obtained by the creation of Telegram Premium gift codes to boost a chat. Each such code boosts the chat 4 times for the duration of the corresponding Telegram Premium subscription.

Source: https://core.telegram.org/bots/api#chatboostsourcegiftcode

```
source: Literal[ChatBoostSourceType.GIFT_CODE]
```

Source of the boost, always 'gift\_code'

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model post init method.

user: User

User for which the gift code was created

# **ChatBoostSourceGiveaway**

The boost was obtained by the creation of a Telegram Premium or a Telegram Star giveaway. This boosts the chat 4 times for the duration of the corresponding Telegram Premium subscription for Telegram Premium giveaways and *prize\_star\_count /* 500 times for one year for Telegram Star giveaways.

Source: https://core.telegram.org/bots/api#chatboostsourcegiveaway

# source: Literal[ChatBoostSourceType.GIVEAWAY]

Source of the boost, always 'giveaway'

# giveaway\_message\_id: int

Identifier of a message in the chat with the giveaway; the message could have been deleted already. May be 0 if the message isn't sent yet.

```
user: User | None
```

Optional. User that won the prize in the giveaway if any; for Telegram Premium giveaways only

None = None, \*\*extra\_data: Any)

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
prize_star_count: int | None
```

*Optional*. The number of Telegram Stars to be split between giveaway winners; for Telegram Star giveaways only

```
is_unclaimed: bool | None
```

Optional. True, if the giveaway was completed, but there was no user to win the prize

## **ChatBoostSourcePremium**

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

Source: https://core.telegram.org/bots/api#chatboostsourcepremium

```
source: Literal[ChatBoostSourceType.PREMIUM]
```

Source of the boost, always 'premium'

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

user: *User* 

User that boosted the chat

## ChatBoostUpdated

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

Source: https://core.telegram.org/bots/api#chatboostupdated

chat: Chat

Chat which was boosted

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model post init method.

boost: ChatBoost

Information about the chat boost

## ChatFullInfo

**class** aiogram.types.chat\_full\_info.**ChatFullInfo**(\*, id: int, type: str, title: str | None = None,

 $username: str \mid None = None, first name: str \mid None$ = None, last\_name: str | None = None, is\_forum: bool | None = None, accent color id: int,  $active\_usernames: List[str] \mid None = None,$ available reactions: List[ReactionTypeEmoji | ReactionTypeCustomEmoji | ReactionTypePaid/ | None = None, background custom emoji id: str | None = None, bio:  $str \mid None = None$ , birthdate: Birthdate | None = None, business intro: BusinessIntro | None = None, business location: BusinessLocation | None = None. business opening hours: BusinessOpeningHours None = None, can\_set\_sticker\_set: bool | None = None, custom\_emoji\_sticker\_set\_name: str | None = *None*, description:  $str \mid None = None$ , emoji\_status\_custom\_emoji\_id: str | None = None, emoji\_status\_expiration\_date: \_datetime\_serializer, return\_type=int, when\_used=unless - none)] | None = None, has\_aggressive\_anti\_spam\_enabled: bool | None = None, has\_hidden\_members: bool | None = None,  $has\_private\_forwards: bool \mid None = None$ , has protected content: bool | None = None, has restricted voice and video messages: bool None = None, has visible history: bool | None =*None*, *invite\_link*: *str* | *None* = *None*, *join\_by\_request*: bool | None = None, join\_to\_send\_messages: bool | None = None,  $linked\_chat\_id$ :  $int \mid None = None$ , *location:* ChatLocation | *None = None*,  $message\_auto\_delete\_time: int | None = None,$ permissions: ChatPermissions | None = None,  $personal\_chat$ : Chat | None = None, photo: ChatPhoto | *None* = *None*, *pinned\_message*: Message | None = None, profile\_accent\_color\_id: int | None = None, profile\_background\_custom\_emoji\_id: str | None = None, slow mode delay: int | None = None,  $sticker\_set\_name: str \mid None = None,$ unrestrict\_boost\_count: int | None = None, max\_reaction\_count: int, can\_send\_paid\_media: bool | None = None, \*\*extra\_data: Any)

This object contains full information about a chat.

Source: https://core.telegram.org/bots/api#chatfullinfo

# id: int

Unique identifier for this chat. This number may have more than 32 significant bits and some programming languages may have difficulty/silent defects in interpreting it. But it has at most 52 significant bits, so a signed 64-bit integer or double-precision float type are safe for storing this identifier.

## type: str

Type of the chat, can be either 'private', 'group', 'supergroup' or 'channel'

# accent\_color\_id: int Identifier of the accent color for the chat name and backgrounds of the chat photo, reply header, and link preview. See accent colors for more details. max\_reaction\_count: int The maximum number of reactions that can be set on a message in the chat title: str | None Optional. Title, for supergroups, channels and group chats username: str | None Optional. Username, for private chats, supergroups and channels if available first\_name: str | None Optional. First name of the other party in a private chat last\_name: str | None Optional. Last name of the other party in a private chat is forum: bool | None Optional. True, if the supergroup chat is a forum (has topics enabled) photo: ChatPhoto | None Optional. Chat photo active\_usernames: List[str] | None Optional. If non-empty, the list of all active chat usernames; for private chats, supergroups and channels birthdate: Birthdate | None Optional. For private chats, the date of birth of the user business\_intro: BusinessIntro | None Optional. For private chats with business accounts, the intro of the business business\_location: BusinessLocation | None Optional. For private chats with business accounts, the location of the business business\_opening\_hours: BusinessOpeningHours | None Optional. For private chats with business accounts, the opening hours of the business personal\_chat: Chat | None Optional. For private chats, the personal channel of the user

available\_reactions: List[ReactionTypeEmoji | ReactionTypeCustomEmoji |

ReactionTypePaid] | None

Optional. List of available reactions allowed in the chat. If omitted, then all emoji reactions are allowed.

background\_custom\_emoji\_id: str | None

Optional. Custom emoji identifier of the emoji chosen by the chat for the reply header and link preview background

profile\_accent\_color\_id: int | None

Optional. Identifier of the accent color for the chat's profile background. See profile accent colors for more details.

profile\_background\_custom\_emoji\_id: str | None

Optional. Custom emoji identifier of the emoji chosen by the chat for its profile background

## emoji\_status\_custom\_emoji\_id: str | None

Optional. Custom emoji identifier of the emoji status of the chat or the other party in a private chat

# emoji\_status\_expiration\_date: DateTime | None

*Optional*. Expiration date of the emoji status of the chat or the other party in a private chat, in Unix time, if any

# model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

# $model_post_init(context: Any, /) \rightarrow None$

We need to both initialize private attributes and call the user-defined model\_post\_init method.

# bio: str | None

Optional. Bio of the other party in a private chat

# has\_private\_forwards: bool | None

*Optional.* True, if privacy settings of the other party in the private chat allows to use tg://user?id=<user\_id> links only in chats with the user

# has\_restricted\_voice\_and\_video\_messages: bool | None

*Optional*. True, if the privacy settings of the other party restrict sending voice and video note messages in the private chat

# join\_to\_send\_messages: bool | None

Optional. True, if users need to join the supergroup before they can send messages

# join\_by\_request: bool | None

*Optional*. True, if all users directly joining the supergroup without using an invite link need to be approved by supergroup administrators

# description: str | None

Optional. Description, for groups, supergroups and channel chats

## invite\_link: str | None

Optional. Primary invite link, for groups, supergroups and channel chats

# pinned\_message: Message | None

Optional. The most recent pinned message (by sending date)

# permissions: ChatPermissions | None

Optional. Default chat member permissions, for groups and supergroups

## can\_send\_paid\_media: bool | None

*Optional*. True, if paid media messages can be sent or forwarded to the channel chat. The field is available only for channel chats.

# slow\_mode\_delay: int | None

*Optional*. For supergroups, the minimum allowed delay between consecutive messages sent by each unprivileged user; in seconds

## unrestrict\_boost\_count: int | None

*Optional*. For supergroups, the minimum number of boosts that a non-administrator user needs to add in order to ignore slow mode and chat permissions

# message\_auto\_delete\_time: int | None

Optional. The time after which all messages sent to the chat will be automatically deleted; in seconds

# has\_aggressive\_anti\_spam\_enabled: bool | None

*Optional*. True, if aggressive anti-spam checks are enabled in the supergroup. The field is only available to chat administrators.

# has\_hidden\_members: bool | None

Optional. True, if non-administrators can only get the list of bots and administrators in the chat

# has\_protected\_content: bool | None

Optional. True, if messages from the chat can't be forwarded to other chats

# has\_visible\_history: bool | None

Optional. True, if new chat members will have access to old messages; available only to chat administrators

#### sticker\_set\_name: str | None

Optional. For supergroups, name of the group sticker set

# can\_set\_sticker\_set: bool | None

Optional. True, if the bot can change the group sticker set

# custom\_emoji\_sticker\_set\_name: str | None

*Optional*. For supergroups, the name of the group's custom emoji sticker set. Custom emoji from this set can be used by all users and bots in the group.

# linked\_chat\_id: int | None

*Optional*. Unique identifier for the linked chat, i.e. the discussion group identifier for a channel and vice versa; for supergroups and channel chats. This identifier may be greater than 32 bits and some programming languages may have difficulty/silent defects in interpreting it. But it is smaller than 52 bits, so a signed 64 bit integer or double-precision float type are safe for storing this identifier.

# location: ChatLocation | None

Optional. For supergroups, the location to which the supergroup is connected

# ChatInviteLink

class aiogram.types.chat\_invite\_link.ChatInviteLink(\*, invite\_link: str, creator: User,

creates\_join\_request: bool, is\_primary: bool, is\_revoked: bool, name: str | None = None, expire\_date: \_datetime\_serializer, return\_type=int, when\_used=unless - none)] | None = None, member\_limit: int | None = None, pending\_join\_request\_count: int | None = None, subscription\_period: int | None = None, subscription\_price: int | None = None, \*\*extra\_data: Any)

Represents an invite link for a chat.

Source: https://core.telegram.org/bots/api#chatinvitelink

# invite\_link: str

The invite link. If the link was created by another chat administrator, then the second part of the link will be replaced with '...'.

creator: User

Creator of the link

## creates\_join\_request: bool

True, if users joining the chat via the link need to be approved by chat administrators

# is\_primary: bool

True, if the link is primary

# is\_revoked: bool

True, if the link is revoked

### name: str | None

Optional. Invite link name

# model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

# expire\_date: DateTime | None

Optional. Point in time (Unix timestamp) when the link will expire or has been expired

# member\_limit: int | None

*Optional*. The maximum number of users that can be members of the chat simultaneously after joining the chat via this invite link; 1-99999

# pending\_join\_request\_count: int | None

Optional. Number of pending join requests created using this link

# subscription\_period: int | None

Optional. The number of seconds the subscription will be active for before the next payment

# subscription\_price: int | None

*Optional*. The amount of Telegram Stars a user must pay initially and after each subsequent subscription period to be a member of the chat using the link

# ChatJoinRequest

Represents a join request sent to a chat.

Source: https://core.telegram.org/bots/api#chatjoinrequest

chat: Chat

Chat to which the request was sent

from\_user: User

User that sent the join request

# user\_chat\_id: int

Identifier of a private chat with the user who sent the join request. This number may have more than 32 significant bits and some programming languages may have difficulty/silent defects in interpreting it. But it has at most 52 significant bits, so a 64-bit integer or double-precision float type are safe for storing this identifier. The bot can use this identifier for 5 minutes to send messages until the join request is processed, assuming no other administrator contacted the user.

# date: DateTime

Date the request was sent in Unix time

## bio: str | None

Optional. Bio of the user.

# invite\_link: ChatInviteLink | None

Optional. Chat invite link that was used by the user to send the join request

```
approve(**kwargs: Any) \rightarrow ApproveChatJoinRequest
```

Shortcut for method aiogram.methods.approve\_chat\_join\_request.ApproveChatJoinRequest will automatically fill method attributes:

- chat\_id
- user\_id

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

Source: https://core.telegram.org/bots/api#approvechatjoinrequest

## Returns

```
instance of method aiogram.methods.approve_chat_join_request.
ApproveChatJoinRequest
```

```
decline(**kwargs: Any) \rightarrow DeclineChatJoinRequest
```

Shortcut for method aiogram.methods.decline\_chat\_join\_request.DeclineChatJoinRequest will automatically fill method attributes:

- chat\_id
- user\_id

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

Source: https://core.telegram.org/bots/api#declinechatjoinrequest

## Returns

```
instance of method aiogram.methods.decline_chat_join_request.
DeclineChatJoinRequest
```

```
answer(text: str, business_connection_id: Optional[str] = None, message_thread_id: Optional[int] = None, parse_mode: Optional[Union[str, Default]] = <Default('parse_mode')>, entities:

Optional[List[MessageEntity]] = None, link_preview_options: Optional[Union[LinkPreviewOptions, Default]] = <Default('link_preview')>, disable_notification: Optional[bool] = None, protect_content: Optional[Union[bool, Default]] = <Default('protect_content')>, message_effect_id: Optional[str] = None, reply_parameters: Optional[ReplyParameters] = None, reply_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None, allow_sending_without_reply: Optional[bool] = None, disable_web_page_preview: Optional[Union[bool, Default]] = <Default('link_preview_is_disabled')>, reply_to_message_id: Optional[int] = None, **kwargs: Any) → SendMessage
```

Shortcut for method <code>aiogram.methods.send\_message.SendMessage</code> will automatically fill method attributes:

• chat\_id

Use this method to send text messages. On success, the sent aiogram.types.message.Message is returned.

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

#### **Parameters**

- text Text of the message to be sent, 1-4096 characters after entities parsing
- **business\_connection\_id** Unique identifier of the business connection on behalf of which the message will be sent
- message\_thread\_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- parse\_mode Mode for parsing entities in the message text. See formatting options for more details.
- **entities** A JSON-serialized list of special entities that appear in message text, which can be specified instead of *parse\_mode*
- link\_preview\_options Link preview generation options for the message
- disable\_notification Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- disable\_web\_page\_preview Disables link previews for links in this message
- reply\_to\_message\_id If the message is a reply, ID of the original message

# **Returns**

instance of method aiogram.methods.send\_message.SendMessage

```
answer_pm(text: str, business_connection_id: Optional[str] = None, message_thread_id: Optional[int] =
None, parse_mode: Optional[Union[str, Default]] = <Default('parse_mode')>, entities:
Optional[List[MessageEntity]] = None, link_preview_options:
Optional[Union[LinkPreviewOptions, Default]] = <Default('link_preview')>,
disable_notification: Optional[bool] = None, protect_content: Optional[Union[bool, Default]] =
<Default('protect_content')>, message_effect_id: Optional[str] = None, reply_parameters:
Optional[ReplyParameters] = None, reply_markup: Optional[Union[InlineKeyboardMarkup,
ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None,
allow_sending_without_reply: Optional[bool] = None, disable_web_page_preview:
Optional[Union[bool, Default]] = <Default('link_preview_is_disabled')>, reply_to_message_id:
Optional[int] = None, **kwargs: Any) → SendMessage
```

Shortcut for method <code>aiogram.methods.send\_message.SendMessage</code> will automatically fill method attributes:

• chat\_id

Use this method to send text messages. On success, the sent aiogram.types.message.Message is returned.

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

## **Parameters**

- text Text of the message to be sent, 1-4096 characters after entities parsing
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- message\_thread\_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- parse\_mode Mode for parsing entities in the message text. See formatting options for more details.
- **entities** A JSON-serialized list of special entities that appear in message text, which can be specified instead of *parse\_mode*
- link\_preview\_options Link preview generation options for the message
- disable\_notification Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_parameters** Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- disable\_web\_page\_preview Disables link previews for links in this message
- reply\_to\_message\_id If the message is a reply, ID of the original message

# **Returns**

instance of method aiogram.methods.send\_message.SendMessage

Shortcut for method aiogram.methods.send\_animation.SendAnimation will automatically fill method attributes:

• chat\_id

Use this method to send animation files (GIF or H.264/MPEG-4 AVC video without sound). On success, the sent aiogram.types.message.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 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. More information on Sending Files »
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- **message\_thread\_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- duration Duration of sent animation in seconds
- width Animation width
- **height** Animation height
- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file\_attach\_name>' if the thumbnail was uploaded using multipart/form-data under <file\_attach\_name>. *More information on Sending Files*
- **caption** Animation caption (may also be used when resending animation by *file\_id*), 0-1024 characters after entities parsing
- parse\_mode Mode for parsing entities in the animation caption. See formatting options for more details.
- **caption\_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse\_mode*

- **show\_caption\_above\_media** Pass True, if the caption must be shown above the message media
- has\_spoiler Pass True if the animation needs to be covered with a spoiler animation
- disable\_notification Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send\_animation.SendAnimation

answer\_animation\_pm(animation: Union[InputFile, str], business\_connection\_id: Optional[str] = None, message\_thread\_id: Optional[int] = None, duration: Optional[int] = None, width: Optional[int] = None, height: Optional[int] = None, thumbnail: Optional[InputFile] = None, caption: Optional[str] = None, parse\_mode: Optional[Union[str, Default]] = <Default('parse\_mode')>, caption\_entities: Optional[List[MessageEntity]] = None, show\_caption\_above\_media: Optional[Union[bool, Default]] = <Default('show\_caption\_above\_media')>, has\_spoiler: Optional[bool] = None, disable\_notification: Optional[bool] = None, protect\_content: Optional[Union[bool, Default]] = <Default('protect\_content')>, message\_effect\_id: Optional[str] = None, reply\_parameters: Optional[ReplyParameters] = None, reply\_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None, allow\_sending\_without\_reply: Optional[bool] = None, reply\_to\_message\_id: Optional[int] = None, \*\*kwargs: Any) → SendAnimation

Shortcut for method aiogram.methods.send\_animation.SendAnimation will automatically fill method attributes:

• chat\_id

Use this method to send animation files (GIF or H.264/MPEG-4 AVC video without sound). On success, the sent *aiogram.types.message.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 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. *More information on Sending Files* »
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent

- message\_thread\_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- duration Duration of sent animation in seconds
- width Animation width
- **height** Animation height
- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file\_attach\_name>' if the thumbnail was uploaded using multipart/form-data under <file\_attach\_name>. *More information on Sending Files*
- **caption** Animation caption (may also be used when resending animation by *file\_id*), 0-1024 characters after entities parsing
- parse\_mode Mode for parsing entities in the animation caption. See formatting options for more details.
- **caption\_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse\_mode*
- **show\_caption\_above\_media** Pass True, if the caption must be shown above the message media
- has\_spoiler Pass True if the animation needs to be covered with a spoiler animation
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send\_animation.SendAnimation

Shortcut for method aiogram.methods.send\_audio.SendAudio will automatically fill method attributes:

# • chat\_id

Use this method to send audio files, if you want Telegram clients to display them in the music player. Your audio must be in the .MP3 or .M4A format. On success, the sent aiogram.types.message.Message is returned. Bots can currently send audio files of up to 50 MB in size, this limit may be changed in the future. For sending voice messages, use the aiogram.methods.send\_voice.SendVoice method instead.

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

### **Parameters**

- audio Audio file to send. Pass a file\_id as String to send an audio file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get an audio file from the Internet, or upload a new one using multipart/form-data. *More information on Sending Files* »
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- **message\_thread\_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- caption Audio caption, 0-1024 characters after entities parsing
- parse\_mode Mode for parsing entities in the audio caption. See formatting options for more details.
- **caption\_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse\_mode*
- duration Duration of the audio in seconds
- **performer** Performer
- title Track name
- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file\_attach\_name>' if the thumbnail was uploaded using multipart/form-data under <file\_attach\_name>. *More information on Sending Files*"""
- disable\_notification Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- **message\_effect\_id** Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_parameters** Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found

• reply\_to\_message\_id - If the message is a reply, ID of the original message

#### Returns

instance of method aiogram.methods.send\_audio.SendAudio

Shortcut for method aiogram.methods.send\_audio.SendAudio will automatically fill method attributes:

chat\_id

Use this method to send audio files, if you want Telegram clients to display them in the music player. Your audio must be in the .MP3 or .M4A format. On success, the sent <code>aiogram.types.message.Message</code> is returned. Bots can currently send audio files of up to 50 MB in size, this limit may be changed in the future. For sending voice messages, use the <code>aiogram.methods.send\_voice.SendVoice</code> method instead.

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

### **Parameters**

- audio Audio file to send. Pass a file\_id as String to send an audio file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get an audio file from the Internet, or upload a new one using multipart/form-data. *More information on Sending Files* »
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- message\_thread\_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **caption** Audio caption, 0-1024 characters after entities parsing
- parse\_mode Mode for parsing entities in the audio caption. See formatting options for more details.
- **caption\_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse\_mode*
- duration Duration of the audio in seconds
- **performer** Performer
- title Track name
- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file\_attach\_name>' if the thumbnail was uploaded

using multipart/form-data under <file\_attach\_name>. More information on Sending Files »

- disable\_notification Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

# Returns

instance of method aiogram.methods.send\_audio.SendAudio

answer\_contact(phone\_number: str, first\_name: str, business\_connection\_id: Optional[str] = None, message\_thread\_id: Optional[int] = None, last\_name: Optional[str] = None, vcard: Optional[str] = None, disable\_notification: Optional[bool] = None, protect\_content: Optional[Union[bool, Default]] = <Default('protect\_content')>, message\_effect\_id: Optional[str] = None, reply\_parameters: Optional[ReplyParameters] = None, reply\_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None, allow\_sending\_without\_reply: Optional[bool] = None, reply\_to\_message\_id: Optional[int] = None, \*\*kwargs: Any) → SendContact

Shortcut for method <code>aiogram.methods.send\_contact.SendContact</code> will automatically fill method attributes:

• chat id

Use this method to send phone contacts. On success, the sent aiogram.types.message.Message is returned.

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

# **Parameters**

- phone\_number Contact's phone number
- **first\_name** Contact's first name
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- **message\_thread\_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- last\_name Contact's last name
- vcard Additional data about the contact in the form of a vCard, 0-2048 bytes
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving

- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send\_contact.SendContact

answer\_contact\_pm(phone\_number: str, first\_name: str, business\_connection\_id: Optional[str] = None, message\_thread\_id: Optional[int] = None, last\_name: Optional[str] = None, vcard: Optional[str] = None, disable\_notification: Optional[bool] = None, protect\_content: Optional[Union[bool, Default]] = <Default('protect\_content')>, message\_effect\_id: Optional[str] = None, reply\_parameters: Optional[ReplyParameters] = None, reply\_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None, allow\_sending\_without\_reply: Optional[bool] = None, reply\_to\_message\_id: Optional[int] = None, \*\*kwargs: Any) → SendContact

Shortcut for method <code>aiogram.methods.send\_contact.SendContact</code> will automatically fill method attributes:

• chat id

Use this method to send phone contacts. On success, the sent aiogram.types.message.Message is returned.

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

# **Parameters**

- **phone\_number** Contact's phone number
- first\_name Contact's first name
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- **message\_thread\_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- last\_name Contact's last name
- vcard Additional data about the contact in the form of a vCard, 0-2048 bytes
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_parameters** Description of the message to reply to

- reply\_markup Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send\_contact.SendContact

Shortcut for method <code>aiogram.methods.send\_document.SendDocument</code> will automatically fill method attributes:

chat\_id

Use this method to send general files. On success, the sent aiogram.types.message.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 File to send. Pass a file\_id as String to send a file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a file from the Internet, or upload a new one using multipart/form-data. More information on Sending Files »
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- **message\_thread\_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file\_attach\_name>' if the thumbnail was uploaded using multipart/form-data under <file\_attach\_name>. *More information on Sending Files*»
- **caption** Document caption (may also be used when resending documents by *file\_id*), 0-1024 characters after entities parsing
- parse\_mode Mode for parsing entities in the document caption. See formatting options for more details.

- **caption\_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse mode*
- **disable\_content\_type\_detection** Disables automatic server-side content type detection for files uploaded using multipart/form-data
- disable\_notification Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_parameters** Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send\_document.SendDocument

Shortcut for method aiogram.methods.send\_document.SendDocument will automatically fill method attributes:

• chat\_id

Use this method to send general files. On success, the sent aiogram.types.message.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** File to send. Pass a file\_id as String to send a file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a file from the Internet, or upload a new one using multipart/form-data. *More information on Sending Files* »
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent

- **message\_thread\_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file\_attach\_name>' if the thumbnail was uploaded using multipart/form-data under <file\_attach\_name>. *More information on Sending Files*
- **caption** Document caption (may also be used when resending documents by *file\_id*), 0-1024 characters after entities parsing
- parse\_mode Mode for parsing entities in the document caption. See formatting options for more details.
- **caption\_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse\_mode*
- **disable\_content\_type\_detection** Disables automatic server-side content type detection for files uploaded using multipart/form-data
- disable\_notification Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

 $instance\ of\ method\ aiogram.methods.send\_document.SendDocument$ 

 $\begin{tabular}{ll} \textbf{answer\_game} (game\_short\_name: str, business\_connection\_id: Optional[str] = None, message\_thread\_id: Optional[int] = None, disable\_notification: Optional[bool] = None, protect\_content: Optional[Union[bool, Default]] = <Default('protect\_content')>, message\_effect\_id: Optional[str] = None, reply\_parameters: Optional[ReplyParameters] = None, reply\_markup: Optional[InlineKeyboardMarkup] = None, allow\_sending\_without\_reply: Optional[bool] = None, reply\_to\_message\_id: Optional[int] = None, **kwargs: Any) \rightarrow SendGame \\ \end{tabular}$ 

Shortcut for method aiogram.methods.send\_game.SendGame will automatically fill method attributes:

• chat id

Use this method to send a game. On success, the sent aiogram.types.message.Message is returned.

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

# **Parameters**

• game\_short\_name — Short name of the game, serves as the unique identifier for the game. Set up your games via @BotFather.

- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- **message\_thread\_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_parameters** Description of the message to reply to
- **reply\_markup** 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.
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send\_game.SendGame

```
answer_game_pm(game_short_name: str, business_connection_id: Optional[str] = None, 
 message_thread_id: Optional[int] = None, disable_notification: Optional[bool] = None, 
 protect_content: Optional[Union[bool, Default]] = \langle Default('protect\_content') \rangle, 
 message_effect_id: Optional[str] = None, reply_parameters: Optional[ReplyParameters] 
 = None, reply_markup: Optional[InlineKeyboardMarkup] = None, 
 allow_sending_without_reply: Optional[bool] = None, reply_to_message_id: 
 Optional[int] = None, **kwargs: Any) → SendGame
```

Shortcut for method aiogram.methods.send\_game.SendGame will automatically fill method attributes:

• chat id

Use this method to send a game. On success, the sent aiogram.types.message.Message is returned.

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

# **Parameters**

- game\_short\_name Short name of the game, serves as the unique identifier for the game. Set up your games via @BotFather.
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- message\_thread\_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** 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.

- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send\_game.SendGame

Shortcut for method <code>aiogram.methods.send\_invoice.SendInvoice</code> will automatically fill method attributes:

• chat\_id

Use this method to send invoices. On success, the sent aiogram.types.message.Message is returned.

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

# **Parameters**

- title Product name, 1-32 characters
- **description** Product description, 1-255 characters
- payload Bot-defined invoice payload, 1-128 bytes. This will not be displayed to the user, use it for your internal processes.
- currency Three-letter ISO 4217 currency code, see more on currencies. Pass 'XTR' for payments in Telegram Stars.
- **prices** Price breakdown, a JSON-serialized list of components (e.g. product price, tax, discount, delivery cost, delivery tax, bonus, etc.). Must contain exactly one item for payments in Telegram Stars.
- message\_thread\_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **provider\_token** Payment provider token, obtained via @BotFather. Pass an empty string for payments in Telegram Stars.
- max\_tip\_amount The maximum accepted amount for tips in the *smallest units* of the currency (integer, **not** float/double). For example, for a maximum tip of US\$ 1.45 pass max\_tip\_amount = 145. See the *exp* parameter in currencies.json, it shows the number of digits past the decimal point for each currency (2 for the majority of currencies). Defaults to 0. Not supported for payments in Telegram Stars.
- **suggested\_tip\_amounts** A JSON-serialized array of suggested amounts of tips in the *smallest units* of the currency (integer, **not** float/double). At most 4 suggested tip amounts

- can be specified. The suggested tip amounts must be positive, passed in a strictly increased order and must not exceed *max tip amount*.
- **start\_parameter** Unique deep-linking parameter. If left empty, **forwarded copies** of the sent message will have a *Pay* button, allowing multiple users to pay directly from the forwarded message, using the same invoice. If non-empty, forwarded copies of the sent message will have a *URL* button with a deep link to the bot (instead of a *Pay* button), with the value used as the start parameter
- provider\_data JSON-serialized data about the invoice, which will be shared with the
  payment provider. A detailed description of required fields should be provided by the payment provider.
- **photo\_url** URL of the product photo for the invoice. Can be a photo of the goods or a marketing image for a service. People like it better when they see what they are paying for.
- **photo\_size** Photo size in bytes
- photo\_width Photo width
- photo\_height Photo height
- **need\_name** Pass True if you require the user's full name to complete the order. Ignored for payments in Telegram Stars.
- **need\_phone\_number** Pass True if you require the user's phone number to complete the order. Ignored for payments in Telegram Stars.
- need\_email Pass True if you require the user's email address to complete the order. Ignored for payments in Telegram Stars.
- **need\_shipping\_address** Pass True if you require the user's shipping address to complete the order. Ignored for payments in Telegram Stars.
- **send\_phone\_number\_to\_provider** Pass True if the user's phone number should be sent to the provider. Ignored for payments in Telegram Stars.
- **send\_email\_to\_provider** Pass True if the user's email address should be sent to the provider. Ignored for payments in Telegram Stars.
- is\_flexible Pass True if the final price depends on the shipping method. Ignored for payments in Telegram Stars.
- disable\_notification Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_parameters** Description of the message to reply to
- **reply\_markup** 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.
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send\_invoice.SendInvoice

```
answer_invoice_pm(title: str, description: str, payload: str, currency: str, prices: List[LabeledPrice],
                      message\_thread\_id: Optional[int] = None, provider\_token: Optional[str] = None,
                      max tip amount: Optional[int] = None, suggested tip amounts: Optional[List[int]] =
                      None, start\_parameter: Optional[str] = None, provider\_data: Optional[str] = None,
                      photo url: Optional[str] = None, photo size: Optional[int] = None, photo width:
                      Optional[int] = None, photo height: Optional[int] = None, need name:
                      Optional[bool] = None, need phone number: Optional[bool] = None, need email:
                      Optional[bool] = None, need\_shipping\_address: Optional[bool] = None,
                      send phone number to provider: Optional[bool] = None, send email to provider:
                      Optional[bool] = None, is\_flexible: Optional[bool] = None, disable\_notification:
                       Optional[bool] = None, protect_content: Optional[Union[bool, Default]] =
                      < Default('protect_content')>, message_effect_id: Optional[str] = None,
                      reply_parameters: Optional[ReplyParameters] = None, reply_markup:
                       Optional[InlineKeyboardMarkup] = None, allow_sending_without_reply:
                      Optional[bool] = None, reply_to_message_id: Optional[int] = None, **kwargs: Any)
                       → SendInvoice
```

Shortcut for method <code>aiogram.methods.send\_invoice.SendInvoice</code> will automatically fill method attributes:

# • chat\_id

Use this method to send invoices. On success, the sent aiogram.types.message.Message is returned.

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

#### Parameters

- title Product name, 1-32 characters
- **description** Product description, 1-255 characters
- **payload** Bot-defined invoice payload, 1-128 bytes. This will not be displayed to the user, use it for your internal processes.
- **currency** Three-letter ISO 4217 currency code, see more on currencies. Pass 'XTR' for payments in Telegram Stars.
- **prices** Price breakdown, a JSON-serialized list of components (e.g. product price, tax, discount, delivery cost, delivery tax, bonus, etc.). Must contain exactly one item for payments in Telegram Stars.
- **message\_thread\_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **provider\_token** Payment provider token, obtained via @BotFather. Pass an empty string for payments in Telegram Stars.
- max\_tip\_amount The maximum accepted amount for tips in the *smallest units* of the currency (integer, **not** float/double). For example, for a maximum tip of US\$ 1.45 pass max\_tip\_amount = 145. See the *exp* parameter in currencies.json, it shows the number of digits past the decimal point for each currency (2 for the majority of currencies). Defaults to 0. Not supported for payments in Telegram Stars.
- **suggested\_tip\_amounts** A JSON-serialized array of suggested amounts of tips in the *smallest units* of the currency (integer, **not** float/double). At most 4 suggested tip amounts can be specified. The suggested tip amounts must be positive, passed in a strictly increased order and must not exceed *max\_tip\_amount*.
- **start\_parameter** Unique deep-linking parameter. If left empty, **forwarded copies** of the sent message will have a *Pay* button, allowing multiple users to pay directly from the forwarded message, using the same invoice. If non-empty, forwarded copies of the sent

- message will have a *URL* button with a deep link to the bot (instead of a *Pay* button), with the value used as the start parameter
- provider\_data JSON-serialized data about the invoice, which will be shared with the
  payment provider. A detailed description of required fields should be provided by the payment provider.
- **photo\_url** URL of the product photo for the invoice. Can be a photo of the goods or a marketing image for a service. People like it better when they see what they are paying for.
- **photo\_size** Photo size in bytes
- photo\_width Photo width
- photo\_height Photo height
- **need\_name** Pass True if you require the user's full name to complete the order. Ignored for payments in Telegram Stars.
- **need\_phone\_number** Pass True if you require the user's phone number to complete the order. Ignored for payments in Telegram Stars.
- need\_email Pass True if you require the user's email address to complete the order. Ignored for payments in Telegram Stars.
- **need\_shipping\_address** Pass True if you require the user's shipping address to complete the order. Ignored for payments in Telegram Stars.
- **send\_phone\_number\_to\_provider** Pass True if the user's phone number should be sent to the provider. Ignored for payments in Telegram Stars.
- **send\_email\_to\_provider** Pass True if the user's email address should be sent to the provider. Ignored for payments in Telegram Stars.
- is\_flexible Pass True if the final price depends on the shipping method. Ignored for payments in Telegram Stars.
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** 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.
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send\_invoice.SendInvoice

Shortcut for method <code>aiogram.methods.send\_location.SendLocation</code> will automatically fill method attributes:

• chat\_id

Use this method to send point on the map. On success, the sent aiogram.types.message.Message is returned.

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

#### **Parameters**

- latitude Latitude of the location
- longitude Longitude of the location
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- message\_thread\_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **horizontal\_accuracy** The radius of uncertainty for the location, measured in meters; 0-1500
- **live\_period** Period in seconds during which the location will be updated (see Live Locations, should be between 60 and 86400, or 0x7FFFFFFF for live locations that can be edited indefinitely.
- **heading** For live locations, a direction in which the user is moving, in degrees. Must be between 1 and 360 if specified.
- **proximity\_alert\_radius** 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 Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found

• reply\_to\_message\_id - If the message is a reply, ID of the original message

#### Returns

instance of method aiogram.methods.send\_location.SendLocation

Shortcut for method <code>aiogram.methods.send\_location</code>. <code>SendLocation</code> will automatically fill method attributes:

• chat\_id

Use this method to send point on the map. On success, the sent aiogram.types.message.Message is returned.

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

#### **Parameters**

- latitude Latitude of the location
- longitude Longitude of the location
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- **message\_thread\_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- horizontal\_accuracy The radius of uncertainty for the location, measured in meters;
   0-1500
- **live\_period** Period in seconds during which the location will be updated (see Live Locations, should be between 60 and 86400, or 0x7FFFFFFF for live locations that can be edited indefinitely.
- **heading** For live locations, a direction in which the user is moving, in degrees. Must be between 1 and 360 if specified.
- proximity\_alert\_radius 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** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to

- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send\_location.SendLocation

Shortcut for method <code>aiogram.methods.send\_media\_group.SendMediaGroup</code> will automatically fill method attributes:

• chat id

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

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

# **Parameters**

- media A JSON-serialized array describing messages to be sent, must include 2-10 items
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- **message\_thread\_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- disable\_notification Sends messages silently. Users will receive a notification with no sound.
- **protect\_content** Protects the contents of the sent messages from forwarding and saving
- **message\_effect\_id** Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_parameters** Description of the message to reply to
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the messages are a reply, ID of the original message

#### Returns

instance of method aiogram.methods.send\_media\_group.SendMediaGroup

Shortcut for method  $aiogram.methods.send\_media\_group.SendMediaGroup$  will automatically fill method attributes:

• chat\_id

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

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

# **Parameters**

- media A JSON-serialized array describing messages to be sent, must include 2-10 items
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- message\_thread\_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- disable\_notification Sends messages silently. Users will receive a notification with no sound.
- **protect\_content** Protects the contents of the sent messages from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the messages are a reply, ID of the original message

# Returns

instance of method aiogram.methods.send\_media\_group.SendMediaGroup

Shortcut for method aiogram.methods.send\_photo.SendPhoto will automatically fill method attributes:

• chat\_id

Use this method to send photos. On success, the sent aiogram. types.message.Message is returned.

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

#### **Parameters**

- **photo** Photo to send. Pass a file\_id as String to send a photo that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a photo from the Internet, or upload a new photo using multipart/form-data. The photo must be at most 10 MB in size. The photo's width and height must not exceed 10000 in total. Width and height ratio must be at most 20. *More information on Sending Files* »
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- message\_thread\_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **caption** Photo caption (may also be used when resending photos by *file\_id*), 0-1024 characters after entities parsing
- parse\_mode Mode for parsing entities in the photo caption. See formatting options for more details.
- **caption\_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse\_mode*
- **show\_caption\_above\_media** Pass True, if the caption must be shown above the message media
- has\_spoiler Pass True if the photo needs to be covered with a spoiler animation
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

# Returns

 $instance\ of\ method\ a \emph{iogram.methods.send\_photo.SendPhoto}$ 

Shortcut for method aiogram.methods.send\_photo.SendPhoto will automatically fill method attributes:

• chat\_id

Use this method to send photos. On success, the sent aiogram. types.message.Message is returned.

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

# **Parameters**

- **photo** Photo to send. Pass a file\_id as String to send a photo that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a photo from the Internet, or upload a new photo using multipart/form-data. The photo must be at most 10 MB in size. The photo's width and height must not exceed 10000 in total. Width and height ratio must be at most 20. *More information on Sending Files* »
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- message\_thread\_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **caption** Photo caption (may also be used when resending photos by *file\_id*), 0-1024 characters after entities parsing
- parse\_mode Mode for parsing entities in the photo caption. See formatting options for more details.
- **caption\_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse\_mode*
- **show\_caption\_above\_media** Pass True, if the caption must be shown above the message media
- has\_spoiler Pass True if the photo needs to be covered with a spoiler animation
- disable\_notification Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- **message\_effect\_id** Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_parameters** Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send\_photo.SendPhoto

```
answer_poll(question: str, options: List[Union[InputPollOption, str]], business_connection_id:

Optional[str] = None, message_thread_id: Optional[int] = None, question_parse_mode:

Optional[Union[str, Default]] = <Default('parse_mode')>, question_entities:

Optional[List[MessageEntity]] = None, is_anonymous: Optional[bool] = None, type:

Optional[str] = None, allows_multiple_answers: Optional[bool] = None, correct_option_id:

Optional[int] = None, explanation: Optional[str] = None, explanation_parse_mode:

Optional[Union[str, Default]] = <Default('parse_mode')>, explanation_entities:

Optional[List[MessageEntity]] = None, open_period: Optional[int] = None, close_date:

Optional[Union[datetime.datetime, datetime.timedelta, int]] = None, is_closed:

Optional[bool] = None, disable_notification: Optional[bool] = None, protect_content:

Optional[Union[bool, Default]] = <Default('protect_content')>, message_effect_id:

Optional[str] = None, reply_parameters: Optional[ReplyParameters] = None, reply_markup:

Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove,
ForceReply]] = None, allow_sending_without_reply: Optional[bool] = None,
reply_to_message_id: Optional[int] = None, **kwargs: Any) → SendPoll
```

Shortcut for method aiogram.methods.send\_poll.SendPoll will automatically fill method attributes:

• chat\_id

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

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

# **Parameters**

- question Poll question, 1-300 characters
- options A JSON-serialized list of 2-10 answer options
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- message\_thread\_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **question\_parse\_mode** Mode for parsing entities in the question. See formatting options for more details. Currently, only custom emoji entities are allowed
- **question\_entities** A JSON-serialized list of special entities that appear in the poll question. It can be specified instead of *question\_parse\_mode*
- is\_anonymous True, if the poll needs to be anonymous, defaults to True
- type Poll type, 'quiz' or 'regular', defaults to 'regular'
- **allows\_multiple\_answers** True, if the poll allows multiple answers, ignored for polls in quiz mode, defaults to False
- correct\_option\_id 0-based identifier of the correct answer option, required for polls in quiz mode
- **explanation** 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** Mode for parsing entities in the explanation. See formatting options for more details.
- **explanation\_entities** A JSON-serialized list of special entities that appear in the poll explanation. It can be specified instead of *explanation\_parse\_mode*
- **open\_period** Amount of time in seconds the poll will be active after creation, 5-600. Can't be used together with *close\_date*.
- **close\_date** 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** Pass True if the poll needs to be immediately closed. This can be useful for poll preview.
- disable\_notification Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send\_poll.SendPoll

**answer\_poll\_pm**(question: str, options: List[Union[InputPollOption, str]], business\_connection\_id:  $Optional[str] = None, message\_thread\_id: Optional[int] = None, question\_parse\_mode:$ *Optional[Union[str, Default]] = <Default('parse\_mode')>, question\_entities:*  $Optional[List[MessageEntity]] = None, is\_anonymous: Optional[bool] = None, type:$  $Optional[str] = None, allows\_multiple\_answers: Optional[bool] = None,$  $correct\_option\_id: Optional[int] = None, explanation: Optional[str] = None,$ explanation\_parse\_mode: Optional[Union[str, Default]] = <Default('parse\_mode')>, explanation\_entities: Optional[List[MessageEntity]] = None, open\_period: Optional[int] = None, close date: Optional[Union[datetime.datetime, datetime.timedelta, int]] = None,  $is\_closed: Optional[bool] = None, disable\_notification: Optional[bool] = None,$ protect content: Optional[Union[bool, Default]] = < Default('protect content')>, message\_effect\_id: Optional[str] = None, reply\_parameters: Optional[ReplyParameters] = None, reply\_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None, allow\_sending\_without\_reply:  $Optional[bool] = None, reply_to_message_id: Optional[int] = None, **kwargs: Any) \rightarrow$ SendPoll

Shortcut for method aiogram.methods.send\_poll.SendPoll will automatically fill method attributes:

• chat\_id

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

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

# **Parameters**

- question Poll question, 1-300 characters
- options A JSON-serialized list of 2-10 answer options
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- **message\_thread\_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **question\_parse\_mode** Mode for parsing entities in the question. See formatting options for more details. Currently, only custom emoji entities are allowed
- **question\_entities** A JSON-serialized list of special entities that appear in the poll question. It can be specified instead of *question\_parse\_mode*
- is\_anonymous True, if the poll needs to be anonymous, defaults to True
- type Poll type, 'quiz' or 'regular', defaults to 'regular'
- allows\_multiple\_answers True, if the poll allows multiple answers, ignored for polls in quiz mode, defaults to False
- correct\_option\_id 0-based identifier of the correct answer option, required for polls in quiz mode
- **explanation** 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** Mode for parsing entities in the explanation. See formatting options for more details.
- **explanation\_entities** A JSON-serialized list of special entities that appear in the poll explanation. It can be specified instead of *explanation\_parse\_mode*
- **open\_period** Amount of time in seconds the poll will be active after creation, 5-600. Can't be used together with *close\_date*.
- **close\_date** 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** Pass True if the poll needs to be immediately closed. This can be useful for poll preview.
- disable\_notification Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- **message\_effect\_id** Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found

• reply\_to\_message\_id - If the message is a reply, ID of the original message

#### Returns

instance of method aiogram.methods.send\_poll.SendPoll

answer\_dice(business\_connection\_id: Optional[str] = None, message\_thread\_id: Optional[int] = None, emoji: Optional[str] = None, disable\_notification: Optional[bool] = None, protect\_content: Optional[Union[bool, Default]] = <Default('protect\_content')>, message\_effect\_id: Optional[str] = None, reply\_parameters: Optional[ReplyParameters] = None, reply\_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None, allow\_sending\_without\_reply: Optional[bool] = None, reply\_to\_message\_id: Optional[int] = None, \*\*kwargs: Any) → SendDice

Shortcut for method <code>aiogram.methods.send\_dice.SendDice</code> will automatically fill method attributes:

• chat\_id

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

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

### **Parameters**

- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- **message\_thread\_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **emoji** 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 Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding
- **message\_effect\_id** Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_parameters** Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

# Returns

instance of method aiogram.methods.send\_dice.SendDice

answer\_dice\_pm(business\_connection\_id: Optional[str] = None, message\_thread\_id: Optional[int] = None, emoji: Optional[str] = None, disable\_notification: Optional[bool] = None, protect\_content: Optional[Union[bool, Default]] = <Default('protect\_content')>, message\_effect\_id: Optional[str] = None, reply\_parameters: Optional[ReplyParameters] = None, reply\_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None, allow\_sending\_without\_reply: Optional[bool] = None, reply\_to\_message\_id: Optional[int] = None, \*\*kwargs: Any) → SendDice

Shortcut for method aiogram.methods.send\_dice.SendDice will automatically fill method attributes:

• chat\_id

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

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

#### **Parameters**

- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- **message\_thread\_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **emoji** 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** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

# Returns

instance of method aiogram.methods.send\_dice.SendDice

Shortcut for method <code>aiogram.methods.send\_sticker.SendSticker</code> will automatically fill method attributes:

• chat\_id

Use this method to send static .WEBP, animated .TGS, or video .WEBM stickers. On success, the sent aiogram.types.message.Message is returned.

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

# **Parameters**

- **sticker** Sticker to send. Pass a file\_id as String to send a file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a .WEBP sticker from the Internet, or upload a new .WEBP, .TGS, or .WEBM sticker using multipart/formdata. *More information on Sending Files* ». Video and animated stickers can't be sent via an HTTP URL.
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- message\_thread\_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- emoji Emoji associated with the sticker; only for just uploaded stickers
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send\_sticker.SendSticker

Shortcut for method aiogram.methods.send\_sticker.SendSticker will automatically fill method attributes:

• chat\_id

Use this method to send static .WEBP, animated .TGS, or video .WEBM stickers. On success, the sent <code>aiogram.types.message.Message</code> is returned.

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

### **Parameters**

• **sticker** – Sticker to send. Pass a file\_id as String to send a file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a .WEBP sticker from the Internet, or upload a new .WEBP, .TGS, or .WEBM sticker using multipart/form-data. *More information on Sending Files* ». Video and animated stickers can't be sent via an HTTP URL.

- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- **message\_thread\_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- emoji Emoji associated with the sticker; only for just uploaded stickers
- disable\_notification Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send\_sticker.SendSticker

answer\_venue(latitude: float, longitude: float, title: str, address: str, business\_connection\_id: Optional[str] = None, message\_thread\_id: Optional[int] = None, foursquare\_id: Optional[str] = None, foursquare\_type: Optional[str] = None, google\_place\_id: Optional[str] = None, google\_place\_type: Optional[str] = None, disable\_notification: Optional[bool] = None, protect\_content: Optional[Union[bool, Default]] = <Default('protect\_content')>, message\_effect\_id: Optional[str] = None, reply\_parameters: Optional[ReplyParameters] = None, reply\_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None, allow\_sending\_without\_reply: Optional[bool] = None, reply\_to\_message\_id: Optional[int] = None, \*\*kwargs: Any) → SendVenue

Shortcut for method aiogram.methods.send\_venue.SendVenue will automatically fill method attributes:

• chat\_id

Use this method to send information about a venue. On success, the sent aiogram.types.message. Message is returned.

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

# **Parameters**

- latitude Latitude of the venue
- longitude Longitude of the venue
- title Name of the venue
- address Address of the venue
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- **message\_thread\_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only

- **foursquare\_id** Foursquare identifier of the venue
- **foursquare\_type** Foursquare type of the venue, if known. (For example, 'arts\_entertainment/default', 'arts\_entertainment/aquarium' or 'food/icecream'.)
- google\_place\_id Google Places identifier of the venue
- **google\_place\_type** Google Places type of the venue. (See supported types.)
- disable\_notification Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send\_venue.SendVenue

```
answer_venue_pm(latitude: float, longitude: float, title: str, address: str, business_connection_id:

Optional[str] = None, message_thread_id: Optional[int] = None, foursquare_id:

Optional[str] = None, foursquare_type: Optional[str] = None, google_place_id:

Optional[str] = None, google_place_type: Optional[str] = None, disable_notification:

Optional[bool] = None, protect_content: Optional[Union[bool, Default]] =

<Default('protect_content')>, message_effect_id: Optional[str] = None,

reply_parameters: Optional[ReplyParameters] = None, reply_markup:

Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove,

ForceReply]] = None, allow_sending_without_reply: Optional[bool] = None,

reply_to_message_id: Optional[int] = None, **kwargs: Any) → SendVenue
```

Shortcut for method aiogram.methods.send\_venue.SendVenue will automatically fill method attributes:

chat\_id

Use this method to send information about a venue. On success, the sent aiogram.types.message. Message is returned.

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

# **Parameters**

- latitude Latitude of the venue
- longitude Longitude of the venue
- title Name of the venue
- address Address of the venue
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent

- **message\_thread\_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **foursquare\_id** Foursquare identifier of the venue
- **foursquare\_type** Foursquare type of the venue, if known. (For example, 'arts\_entertainment/default', 'arts\_entertainment/aquarium' or 'food/icecream'.)
- google\_place\_id Google Places identifier of the venue
- **google\_place\_type** Google Places type of the venue. (See supported types.)
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send\_venue.SendVenue

Shortcut for method aiogram.methods.send\_video.SendVideo will automatically fill method attributes:

• chat\_id

Use this method to send video files, Telegram clients support MPEG4 videos (other formats may be sent as aiogram.types.document.Document). On success, the sent aiogram.types.message.Message is returned. Bots can currently send video files of up to 50 MB in size, this limit may be changed in the future.

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

# **Parameters**

• **video** – Video to send. Pass a file\_id as String to send a video that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a video from the

Internet, or upload a new video using multipart/form-data. *More information on Sending Files* »

- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- **message\_thread\_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- duration Duration of sent video in seconds
- width Video width
- height Video height
- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file\_attach\_name>' if the thumbnail was uploaded using multipart/form-data under <file\_attach\_name>. *More information on Sending Files*"""
- **caption** Video caption (may also be used when resending videos by *file\_id*), 0-1024 characters after entities parsing
- parse\_mode Mode for parsing entities in the video caption. See formatting options for more details.
- **caption\_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse\_mode*
- **show\_caption\_above\_media** Pass True, if the caption must be shown above the message media
- has\_spoiler Pass True if the video needs to be covered with a spoiler animation
- **supports\_streaming** Pass True if the uploaded video is suitable for streaming
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

# Returns

instance of method aiogram.methods.send\_video.SendVideo

model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

 $model_post_init(context: Any, /) \rightarrow None$ 

We need to both initialize private attributes and call the user-defined model\_post\_init method.

Shortcut for method aiogram.methods.send\_video.SendVideo will automatically fill method attributes:

• chat\_id

Use this method to send video files, Telegram clients support MPEG4 videos (other formats may be sent as aiogram.types.document.Document). On success, the sent aiogram.types.message.Message is returned. Bots can currently send video files of up to 50 MB in size, this limit may be changed in the future.

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

## **Parameters**

- video Video to send. Pass a file\_id as String to send a video that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a video from the Internet, or upload a new video using multipart/form-data. More information on Sending Files »
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- message\_thread\_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- duration Duration of sent video in seconds
- width Video width
- **height** Video height
- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file\_attach\_name>' if the thumbnail was uploaded using multipart/form-data under <file\_attach\_name>. *More information on Sending Files*
- **caption** Video caption (may also be used when resending videos by *file\_id*), 0-1024 characters after entities parsing

- parse\_mode Mode for parsing entities in the video caption. See formatting options for more details.
- **caption\_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse\_mode*
- **show\_caption\_above\_media** Pass True, if the caption must be shown above the message media
- has\_spoiler Pass True if the video needs to be covered with a spoiler animation
- **supports\_streaming** Pass True if the uploaded video is suitable for streaming
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send\_video.SendVideo

Shortcut for method aiogram.methods.send\_video\_note.SendVideoNote will automatically fill method attributes:

• chat\_id

As of v.4.0, Telegram clients support rounded square MPEG4 videos of up to 1 minute long. Use this method to send video messages. On success, the sent aiogram.types.message.Message is returned.

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

## **Parameters**

video\_note – Video note to send. Pass a file\_id as String to send a video note that exists
on the Telegram servers (recommended) or upload a new video using multipart/form-data.
 *More information on Sending Files* ». Sending video notes by a URL is currently unsupported

- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- **message\_thread\_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- duration Duration of sent video in seconds
- length Video width and height, i.e. diameter of the video message
- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file\_attach\_name>' if the thumbnail was uploaded using multipart/form-data under <file\_attach\_name>. *More information on Sending Files*""
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send\_video\_note.SendVideoNote

```
answer_video_note_pm(video_note: Union[InputFile, str], business_connection_id: Optional[str] = None, message_thread_id: Optional[int] = None, duration: Optional[int] = None, length: Optional[int] = None, thumbnail: Optional[InputFile] = None, disable_notification: Optional[bool] = None, protect_content:

Optional[Union[bool, Default]] = <Default('protect_content')>, message_effect_id: Optional[str] = None, reply_parameters:

Optional[ReplyParameters] = None, reply_markup:
Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None, allow_sending_without_reply:
Optional[bool] = None, reply_to_message_id: Optional[int] = None, **kwargs: Any) → SendVideoNote
```

Shortcut for method <code>aiogram.methods.send\_video\_note.SendVideoNote</code> will automatically fill method attributes:

• chat id

As of v.4.0, Telegram clients support rounded square MPEG4 videos of up to 1 minute long. Use this method to send video messages. On success, the sent aiogram.types.message.Message is returned.

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

# **Parameters**

- **video\_note** Video note to send. Pass a file\_id as String to send a video note that exists on the Telegram servers (recommended) or upload a new video using multipart/form-data. *More information on Sending Files* ». Sending video notes by a URL is currently unsupported
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- **message\_thread\_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- duration Duration of sent video in seconds
- length Video width and height, i.e. diameter of the video message
- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file\_attach\_name>' if the thumbnail was uploaded using multipart/form-data under <file\_attach\_name>. *More information on Sending Files*""
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

 $instance\ of\ method\ aiogram.methods.send\_video\_note.SendVideoNote$ 

Shortcut for method aiogram.methods.send\_voice.SendVoice will automatically fill method attributes:

• chat\_id

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, or in .MP3 format,

or in .M4A format (other formats may be sent as aiogram.types.audio.Audio or aiogram.types.document.Document). On success, the sent aiogram.types.message.Message is returned. Bots can currently send voice messages of up to 50 MB in size, this limit may be changed in the future.

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

# **Parameters**

- **voice** Audio file to send. Pass a file\_id as String to send a file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a file from the Internet, or upload a new one using multipart/form-data. *More information on Sending Files* »
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- message\_thread\_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- caption Voice message caption, 0-1024 characters after entities parsing
- parse\_mode Mode for parsing entities in the voice message caption. See formatting options for more details.
- **caption\_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse\_mode*
- duration Duration of the voice message in seconds
- disable\_notification Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

### Returns

instance of method aiogram.methods.send\_voice.SendVoice

Shortcut for method aiogram.methods.send\_voice.SendVoice will automatically fill method attributes:

#### • chat\_id

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, or in .MP3 format, or in .M4A format (other formats may be sent as <code>aiogram.types.audio.Audio</code> or <code>aiogram.types.document.Document</code>). On success, the sent <code>aiogram.types.message</code> is returned. Bots can currently send voice messages of up to 50 MB in size, this limit may be changed in the future.

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

# **Parameters**

- **voice** Audio file to send. Pass a file\_id as String to send a file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a file from the Internet, or upload a new one using multipart/form-data. *More information on Sending Files* »
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- message\_thread\_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- caption Voice message caption, 0-1024 characters after entities parsing
- parse\_mode Mode for parsing entities in the voice message caption. See formatting options for more details.
- **caption\_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse mode*
- duration Duration of the voice message in seconds
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_parameters** Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

# Returns

instance of method aiogram.methods.send\_voice.SendVoice

# ChatLocation

Represents a location to which a chat is connected.

Source: https://core.telegram.org/bots/api#chatlocation

location: Location

The location to which the supergroup is connected. Can't be a live location.

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

address: str

Location address; 1-64 characters, as defined by the chat owner

# ChatMember

```
class aiogram.types.chat_member.ChatMember(**extra_data: Any)
```

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

- aiogram.types.chat\_member\_owner.ChatMemberOwner
- aiogram.types.chat\_member\_administrator.ChatMemberAdministrator
- aiogram.types.chat\_member\_member.ChatMemberMember
- aiogram.types.chat\_member\_restricted.ChatMemberRestricted
- aiogram.types.chat\_member\_left.ChatMemberLeft
- aiogram.types.chat\_member\_banned.ChatMemberBanned

Source: https://core.telegram.org/bots/api#chatmember

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

# ChatMemberAdministrator

class aiogram.types.chat\_member\_administrator.ChatMemberAdministrator(\*, status: Lit-

 $eral [{\it Chat Member Status. ADMINISTRATOR}]$ 

= ChatMemberStatus.ADMINISTRATOR,

user: User,

can\_be\_edited: bool, is\_anonymous: bool, can\_manage\_chat: bool, can\_delete\_messages:

bool,

can\_manage\_video\_chats:

bool,

can\_restrict\_members:

bool,

can\_promote\_members: bool, can\_change\_info: bool, can\_invite\_users:

bool, can\_post\_stories:

bool, can\_edit\_stories: bool, can\_delete\_stories:

bool,

 $can\_post\_messages:$ 

 $bool \mid None = None,$ 

can\_edit\_messages: bool

| None = None,

can\_pin\_messages: bool

| None = None, can\_manage\_topics: bool | None = None, custom\_title: str | None = None, \*\*extra\_data:

Any)

Represents a chat member that has some additional privileges.

Source: https://core.telegram.org/bots/api#chatmemberadministrator

# status: Literal[ChatMemberStatus.ADMINISTRATOR]

The member's status in the chat, always 'administrator'

user: *User* 

Information about the user

can\_be\_edited: bool

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

is\_anonymous: bool

True, if the user's presence in the chat is hidden

can\_manage\_chat: bool

True, if the administrator can access the chat event log, get boost list, see hidden supergroup and channel members, report spam messages and ignore slow mode. Implied by any other administrator privilege.

can\_delete\_messages: bool

True, if the administrator can delete messages of other users

## can\_manage\_video\_chats: bool

True, if the administrator can manage video chats

# can\_restrict\_members: bool

True, if the administrator can restrict, ban or unban chat members, or access supergroup statistics

#### can\_promote\_members: bool

True, if the administrator can add new administrators with a subset of their own privileges or demote administrators that they have promoted, directly or indirectly (promoted by administrators that were appointed by the user)

# can\_change\_info: bool

True, if the user is allowed to change the chat title, photo and other settings

## model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

# $model_post_init(context: Any, /) \rightarrow None$

We need to both initialize private attributes and call the user-defined model\_post\_init method.

# can\_invite\_users: bool

True, if the user is allowed to invite new users to the chat

### can\_post\_stories: bool

True, if the administrator can post stories to the chat

### can\_edit\_stories: bool

True, if the administrator can edit stories posted by other users, post stories to the chat page, pin chat stories, and access the chat's story archive

# can\_delete\_stories: bool

True, if the administrator can delete stories posted by other users

## can\_post\_messages: bool | None

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

# can\_edit\_messages: bool | None

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

### can\_pin\_messages: bool | None

Optional. True, if the user is allowed to pin messages; for groups and supergroups only

# can\_manage\_topics: bool | None

Optional. True, if the user is allowed to create, rename, close, and reopen forum topics; for supergroups only

# custom\_title: str | None

Optional. Custom title for this user

### **ChatMemberBanned**

class aiogram.types.chat\_member\_banned.ChatMemberBanned(\*, status:

Literal[ChatMemberStatus.KICKED] = ChatMemberStatus.KICKED, user: User, until\_date: \_datetime\_serializer, return\_type=int, when\_used=unless - none)], \*\*extra\_data: Any)

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

Source: https://core.telegram.org/bots/api#chatmemberbanned

# status: Literal[ChatMemberStatus.KICKED]

The member's status in the chat, always 'kicked'

user: *User* 

Information about the user

# model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

## until\_date: DateTime

Date when restrictions will be lifted for this user; Unix time. If 0, then the user is banned forever

## ChatMemberLeft

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

Source: https://core.telegram.org/bots/api#chatmemberleft

# status: Literal[ChatMemberStatus.LEFT]

The member's status in the chat, always 'left'

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

user: *User* 

Information about the user

### ChatMemberMember

Literal[ChatMemberStatus.MEMBER] = ChatMemberStatus.MEMBER, user: User, until\_date: \_datetime\_serializer, return\_type=int, when\_used=unless - none)] | None = None, \*\*extra\_data: Any)

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

Source: https://core.telegram.org/bots/api#chatmembermember

# status: Literal[ChatMemberStatus.MEMBER]

The member's status in the chat, always 'member'

user: *User* 

Information about the user

# model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

## until\_date: DateTime | None

Optional. Date when the user's subscription will expire; Unix time

## **ChatMemberOwner**

class aiogram.types.chat\_member\_owner.ChatMemberOwner(\*, status:

Literal[ChatMemberStatus.CREATOR] =
ChatMemberStatus.CREATOR, user: User,
is\_anonymous: bool, custom\_title: str | None
= None, \*\*extra\_data: Any)

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

Source: https://core.telegram.org/bots/api#chatmemberowner

# status: Literal[ChatMemberStatus.CREATOR]

The member's status in the chat, always 'creator'

user: *User* 

Information about the user

# model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

is\_anonymous: bool

True, if the user's presence in the chat is hidden

custom\_title: str | None

Optional. Custom title for this user

### ChatMemberRestricted

class aiogram.types.chat\_member\_restricted.ChatMemberRestricted(\*, status: Literal[ChatMemberStatus.RESTRICTED] = ChatMemberStatus.RESTRICTED, user: User, is\_member: bool, can send messages: bool, can send audios: bool, can send documents: bool, can\_send\_photos: bool, can\_send\_videos: bool, can send video notes: bool, can\_send\_voice\_notes: bool, can send polls: bool, can\_send\_other\_messages: bool, can\_add\_web\_page\_previews: bool, can\_change\_info: bool, can\_invite\_users: bool, can\_pin\_messages: bool, can\_manage\_topics: bool, until\_date: \_datetime\_serializer, return\_type=int,

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

Source: https://core.telegram.org/bots/api#chatmemberrestricted

# status: Literal[ChatMemberStatus.RESTRICTED]

The member's status in the chat, always 'restricted'

user: *User* 

Information about the user

is member: bool

True, if the user is a member of the chat at the moment of the request

can\_send\_messages: bool

True, if the user is allowed to send text messages, contacts, giveaways, giveaway winners, invoices, locations and venues

can\_send\_audios: bool

True, if the user is allowed to send audios

can\_send\_documents: bool

True, if the user is allowed to send documents

can\_send\_photos: bool

True, if the user is allowed to send photos

can\_send\_videos: bool

True, if the user is allowed to send videos

can\_send\_video\_notes: bool

True, if the user is allowed to send video notes

when\_used=unless - none)],

\*\*extra data: Any)

# model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

# $model_post_init(context: Any, /) \rightarrow None$

We need to both initialize private attributes and call the user-defined model\_post\_init method.

# can\_send\_voice\_notes: bool

True, if the user is allowed to send voice notes

### can\_send\_polls: bool

True, if the user is allowed to send polls

### can\_send\_other\_messages: bool

True, if the user is allowed to send animations, games, stickers and use inline bots

# can\_add\_web\_page\_previews: bool

True, if the user is allowed to add web page previews to their messages

# can\_change\_info: bool

True, if the user is allowed to change the chat title, photo and other settings

### can\_invite\_users: bool

True, if the user is allowed to invite new users to the chat

# can\_pin\_messages: bool

True, if the user is allowed to pin messages

### can\_manage\_topics: bool

True, if the user is allowed to create forum topics

# until\_date: DateTime

Date when restrictions will be lifted for this user; Unix time. If 0, then the user is restricted forever

# ChatMemberUpdated

**class** aiogram.types.chat\_member\_updated.**ChatMemberUpdated**(\*, chat: Chat, from\_user: User, date:

\_datetime\_serializer, return\_type=int,

when\_used=unless - none)],

old\_chat\_member: ChatMemberOwner |

ChatMemberAdministrator |

ChatMemberMember |

ChatMemberRestricted |

ChatMemberLeft | ChatMemberBanned,

new\_chat\_member: ChatMemberOwner

| ChatMemberAdministrator |

ChatMember I

ChatMemberRestricted |

ChatMemberLeft | ChatMemberBanned,

invite\_link: ChatInviteLink | None =

None, via\_join\_request: bool | None =

None, via\_chat\_folder\_invite\_link: bool |

 $None = None, **extra_data: Any)$ 

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

Source: https://core.telegram.org/bots/api#chatmemberupdated

chat: Chat

Chat the user belongs to

from\_user: User

Performer of the action, which resulted in the change

date: DateTime

Date the change was done in Unix time

old\_chat\_member: ChatMemberOwner | ChatMemberAdministrator | ChatMemberMember |
ChatMemberRestricted | ChatMemberLeft | ChatMemberBanned

Previous information about the chat member

new\_chat\_member: ChatMemberOwner | ChatMemberAdministrator | ChatMemberMember |
ChatMemberRestricted | ChatMemberLeft | ChatMemberBanned

New information about the chat member

```
invite_link: ChatInviteLink | None
```

Optional. Chat invite link, which was used by the user to join the chat; for joining by invite link events only.

# via\_join\_request: bool | None

*Optional*. True, if the user joined the chat after sending a direct join request without using an invite link and being approved by an administrator

### via\_chat\_folder\_invite\_link: bool | None

Optional. True, if the user joined the chat via a chat folder invite link

answer(text: str, business\_connection\_id: Optional[str] = None, message\_thread\_id: Optional[int] = None, parse\_mode: Optional[Union[str, Default]] = <Default('parse\_mode')>, entities:

Optional[List[MessageEntity]] = None, link\_preview\_options: Optional[Union[LinkPreviewOptions, Default]] = <Default('link\_preview')>, disable\_notification: Optional[bool] = None, protect\_content: Optional[Union[bool, Default]] = <Default('protect\_content')>, message\_effect\_id: Optional[str] = None, reply\_parameters: Optional[ReplyParameters] = None, reply\_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None, allow\_sending\_without\_reply: Optional[bool] = None, disable\_web\_page\_preview: Optional[Union[bool, Default]] = <Default('link\_preview\_is\_disabled')>, reply\_to\_message\_id: Optional[int] = None, \*\*kwargs: Any) → SendMessage

Shortcut for method <code>aiogram.methods.send\_message.SendMessage</code> will automatically fill method attributes:

• chat\_id

Use this method to send text messages. On success, the sent aiogram.types.message.Message is returned.

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

#### **Parameters**

- text Text of the message to be sent, 1-4096 characters after entities parsing
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- **message\_thread\_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only

- parse\_mode Mode for parsing entities in the message text. See formatting options for more details.
- **entities** A JSON-serialized list of special entities that appear in message text, which can be specified instead of *parse\_mode*
- link\_preview\_options Link preview generation options for the message
- disable\_notification Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- disable\_web\_page\_preview Disables link previews for links in this message
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send\_message.SendMessage

```
answer_animation (animation: Union[InputFile, str], business_connection_id: Optional[str] = None, message_thread_id: Optional[int] = None, duration: Optional[int] = None, width: Optional[int] = None, height: Optional[int] = None, thumbnail: Optional[InputFile] = None, caption: Optional[str] = None, parse_mode: Optional[Union[str, Default]] = < Default('parse_mode') >, caption_entities: Optional[List[MessageEntity]] = None, show_caption_above_media: Optional[Union[bool, Default]] = < Default('show_caption_above_media') >, has_spoiler: Optional[bool] = None, disable_notification: Optional[bool] = None, protect_content: Optional[Union[bool, Default]] = < Default('protect_content') >, message_effect_id: Optional[str] = None, reply_parameters: Optional[ReplyParameters] = None, reply_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None, allow_sending_without_reply: Optional[bool] = None, reply_to_message_id: Optional[int] = None, **kwargs: Any) → SendAnimation
```

Shortcut for method aiogram.methods.send\_animation.SendAnimation will automatically fill method attributes:

chat\_id

Use this method to send animation files (GIF or H.264/MPEG-4 AVC video without sound). On success, the sent <code>aiogram.types.message.Message</code> 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 – 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. *More information on Sending Files »* 

- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- message\_thread\_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- duration Duration of sent animation in seconds
- width Animation width
- height Animation height
- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file\_attach\_name>' if the thumbnail was uploaded using multipart/form-data under <file\_attach\_name>. *More information on Sending Files*""
- **caption** Animation caption (may also be used when resending animation by *file\_id*), 0-1024 characters after entities parsing
- parse\_mode Mode for parsing entities in the animation caption. See formatting options for more details.
- **caption\_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse\_mode*
- **show\_caption\_above\_media** Pass True, if the caption must be shown above the message media
- has\_spoiler Pass True if the animation needs to be covered with a spoiler animation
- disable\_notification Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

### Returns

instance of method aiogram.methods.send\_animation.SendAnimation

Shortcut for method <code>aiogram.methods.send\_audio.SendAudio</code> will automatically fill method attributes:

#### • chat id

Use this method to send audio files, if you want Telegram clients to display them in the music player. Your audio must be in the .MP3 or .M4A format. On success, the sent <code>aiogram.types.message.Message</code> is returned. Bots can currently send audio files of up to 50 MB in size, this limit may be changed in the future. For sending voice messages, use the <code>aiogram.methods.send\_voice.SendVoice</code> method instead.

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

#### **Parameters**

- audio Audio file to send. Pass a file\_id as String to send an audio file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get an audio file from the Internet, or upload a new one using multipart/form-data. *More information on Sending Files* »
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- message\_thread\_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- caption Audio caption, 0-1024 characters after entities parsing
- parse\_mode Mode for parsing entities in the audio caption. See formatting options for more details.
- **caption\_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse\_mode*
- duration Duration of the audio in seconds
- **performer** Performer
- title Track name
- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file\_attach\_name>' if the thumbnail was uploaded using multipart/form-data under <file\_attach\_name>. *More information on Sending Files*
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving

- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send\_audio.SendAudio

answer\_contact(phone\_number: str, first\_name: str, business\_connection\_id: Optional[str] = None, message\_thread\_id: Optional[int] = None, last\_name: Optional[str] = None, vcard: Optional[str] = None, disable\_notification: Optional[bool] = None, protect\_content: Optional[Union[bool, Default]] = <Default('protect\_content')>, message\_effect\_id: Optional[str] = None, reply\_parameters: Optional[ReplyParameters] = None, reply\_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None, allow\_sending\_without\_reply: Optional[bool] = None, reply\_to\_message\_id: Optional[int] = None, \*\*kwargs: Any) → SendContact

Shortcut for method <code>aiogram.methods.send\_contact</code>. <code>SendContact</code> will automatically fill method attributes:

• chat id

Use this method to send phone contacts. On success, the sent aiogram.types.message.Message is returned.

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

## **Parameters**

- **phone\_number** Contact's phone number
- first\_name Contact's first name
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- **message\_thread\_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- last\_name Contact's last name
- vcard Additional data about the contact in the form of a vCard, 0-2048 bytes
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_parameters** Description of the message to reply to

- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send\_contact.SendContact

Shortcut for method <code>aiogram.methods.send\_document.SendDocument</code> will automatically fill method attributes:

chat\_id

Use this method to send general files. On success, the sent aiogram.types.message.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 File to send. Pass a file\_id as String to send a file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a file from the Internet, or upload a new one using multipart/form-data. *More information on Sending Files* »
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- **message\_thread\_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file\_attach\_name>' if the thumbnail was uploaded using multipart/form-data under <file\_attach\_name>. *More information on Sending Files*
- **caption** Document caption (may also be used when resending documents by *file\_id*), 0-1024 characters after entities parsing
- parse\_mode Mode for parsing entities in the document caption. See formatting options for more details.

- **caption\_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse mode*
- **disable\_content\_type\_detection** Disables automatic server-side content type detection for files uploaded using multipart/form-data
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_parameters** Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send\_document.SendDocument

 $\begin{tabular}{ll} \textbf{answer\_game} (game\_short\_name: str, business\_connection\_id: Optional[str] = None, message\_thread\_id: Optional[int] = None, disable\_notification: Optional[bool] = None, protect\_content: Optional[Union[bool, Default]] = <Default('protect\_content')>, message\_effect\_id: Optional[str] = None, reply\_parameters: Optional[ReplyParameters] = None, reply\_markup: Optional[InlineKeyboardMarkup] = None, allow\_sending\_without\_reply: Optional[bool] = None, reply\_to\_message\_id: Optional[int] = None, **kwargs: Any) \rightarrow SendGame \\ \end{tabular}$ 

Shortcut for method aiogram.methods.send\_game.SendGame will automatically fill method attributes:

• chat\_id

Use this method to send a game. On success, the sent aiogram.types.message.Message is returned.

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

# **Parameters**

- game\_short\_name Short name of the game, serves as the unique identifier for the game. Set up your games via @BotFather.
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- message\_thread\_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** 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.

- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send\_game.SendGame

Shortcut for method <code>aiogram.methods.send\_invoice.SendInvoice</code> will automatically fill method attributes:

• chat\_id

Use this method to send invoices. On success, the sent aiogram.types.message.Message is returned.

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

## **Parameters**

- title Product name, 1-32 characters
- **description** Product description, 1-255 characters
- **payload** Bot-defined invoice payload, 1-128 bytes. This will not be displayed to the user, use it for your internal processes.
- **currency** Three-letter ISO 4217 currency code, see more on currencies. Pass 'XTR' for payments in Telegram Stars.
- **prices** Price breakdown, a JSON-serialized list of components (e.g. product price, tax, discount, delivery cost, delivery tax, bonus, etc.). Must contain exactly one item for payments in Telegram Stars.
- message\_thread\_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **provider\_token** Payment provider token, obtained via @BotFather. Pass an empty string for payments in Telegram Stars.
- max\_tip\_amount The maximum accepted amount for tips in the *smallest units* of the currency (integer, **not** float/double). For example, for a maximum tip of US\$ 1.45 pass max\_tip\_amount = 145. See the *exp* parameter in currencies.json, it shows the number of digits past the decimal point for each currency (2 for the majority of currencies). Defaults to 0. Not supported for payments in Telegram Stars.
- **suggested\_tip\_amounts** A JSON-serialized array of suggested amounts of tips in the *smallest units* of the currency (integer, **not** float/double). At most 4 suggested tip amounts

- can be specified. The suggested tip amounts must be positive, passed in a strictly increased order and must not exceed *max tip amount*.
- **start\_parameter** Unique deep-linking parameter. If left empty, **forwarded copies** of the sent message will have a *Pay* button, allowing multiple users to pay directly from the forwarded message, using the same invoice. If non-empty, forwarded copies of the sent message will have a *URL* button with a deep link to the bot (instead of a *Pay* button), with the value used as the start parameter
- provider\_data JSON-serialized data about the invoice, which will be shared with the
  payment provider. A detailed description of required fields should be provided by the payment provider.
- **photo\_url** URL of the product photo for the invoice. Can be a photo of the goods or a marketing image for a service. People like it better when they see what they are paying for.
- **photo\_size** Photo size in bytes
- photo\_width Photo width
- photo\_height Photo height
- **need\_name** Pass True if you require the user's full name to complete the order. Ignored for payments in Telegram Stars.
- **need\_phone\_number** Pass True if you require the user's phone number to complete the order. Ignored for payments in Telegram Stars.
- need\_email Pass True if you require the user's email address to complete the order. Ignored for payments in Telegram Stars.
- **need\_shipping\_address** Pass True if you require the user's shipping address to complete the order. Ignored for payments in Telegram Stars.
- **send\_phone\_number\_to\_provider** Pass True if the user's phone number should be sent to the provider. Ignored for payments in Telegram Stars.
- **send\_email\_to\_provider** Pass True if the user's email address should be sent to the provider. Ignored for payments in Telegram Stars.
- **is\_flexible** Pass True if the final price depends on the shipping method. Ignored for payments in Telegram Stars.
- disable\_notification Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_parameters** Description of the message to reply to
- **reply\_markup** 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.
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send\_invoice.SendInvoice

Shortcut for method <code>aiogram.methods.send\_location.SendLocation</code> will automatically fill method attributes:

• chat\_id

Use this method to send point on the map. On success, the sent aiogram.types.message.Message is returned.

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

#### **Parameters**

- latitude Latitude of the location
- longitude Longitude of the location
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- message\_thread\_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **horizontal\_accuracy** The radius of uncertainty for the location, measured in meters; 0-1500
- **live\_period** Period in seconds during which the location will be updated (see Live Locations, should be between 60 and 86400, or 0x7FFFFFFF for live locations that can be edited indefinitely.
- **heading** For live locations, a direction in which the user is moving, in degrees. Must be between 1 and 360 if specified.
- **proximity\_alert\_radius** 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 Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found

• reply\_to\_message\_id - If the message is a reply, ID of the original message

#### Returns

instance of method aiogram.methods.send\_location.SendLocation

Shortcut for method  $aiogram.methods.send\_media\_group.SendMediaGroup$  will automatically fill method attributes:

• chat\_id

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

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

#### **Parameters**

- media A JSON-serialized array describing messages to be sent, must include 2-10 items
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- **message\_thread\_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **disable\_notification** Sends messages silently. Users will receive a notification with no sound.
- **protect\_content** Protects the contents of the sent messages from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_parameters** Description of the message to reply to
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the messages are a reply, ID of the original message

#### Returns

instance of method aiogram.methods.send\_media\_group.SendMediaGroup

Shortcut for method <code>aiogram.methods.send\_photo.SendPhoto</code> will automatically fill method attributes:

• chat id

Use this method to send photos. On success, the sent aiogram.types.message.Message is returned.

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

#### **Parameters**

- **photo** Photo to send. Pass a file\_id as String to send a photo that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a photo from the Internet, or upload a new photo using multipart/form-data. The photo must be at most 10 MB in size. The photo's width and height must not exceed 10000 in total. Width and height ratio must be at most 20. *More information on Sending Files* »
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- message\_thread\_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **caption** Photo caption (may also be used when resending photos by *file\_id*), 0-1024 characters after entities parsing
- parse\_mode Mode for parsing entities in the photo caption. See formatting options for more details.
- **caption\_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse\_mode*
- **show\_caption\_above\_media** Pass True, if the caption must be shown above the message media
- has\_spoiler Pass True if the photo needs to be covered with a spoiler animation
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- **message\_effect\_id** Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send\_photo.SendPhoto

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
answer_poll(question: str, options: List[Union[InputPollOption, str]], business_connection_id:

Optional[str] = None, message_thread_id: Optional[int] = None, question_parse_mode:

Optional[Union[str, Default]] = <Default('parse_mode')>, question_entities:

Optional[List[MessageEntity]] = None, is_anonymous: Optional[bool] = None, type:

Optional[str] = None, allows_multiple_answers: Optional[bool] = None, correct_option_id:

Optional[int] = None, explanation: Optional[str] = None, explanation_parse_mode:

Optional[Union[str, Default]] = <Default('parse_mode')>, explanation_entities:

Optional[List[MessageEntity]] = None, open_period: Optional[int] = None, close_date:

Optional[Union[datetime.datetime, datetime.timedelta, int]] = None, is_closed:

Optional[bool] = None, disable_notification: Optional[bool] = None, protect_content:

Optional[Union[bool, Default]] = <Default('protect_content')>, message_effect_id:

Optional[str] = None, reply_parameters: Optional[ReplyParameters] = None, reply_markup:

Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove,
ForceReply]] = None, allow_sending_without_reply: Optional[bool] = None,
reply_to_message_id: Optional[int] = None, **kwargs: Any) → SendPoll
```

Shortcut for method aiogram.methods.send\_poll.SendPoll will automatically fill method attributes:

• chat\_id

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

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

## **Parameters**

- question Poll question, 1-300 characters
- options A JSON-serialized list of 2-10 answer options
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- message\_thread\_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **question\_parse\_mode** Mode for parsing entities in the question. See formatting options for more details. Currently, only custom emoji entities are allowed
- **question\_entities** A JSON-serialized list of special entities that appear in the poll question. It can be specified instead of *question\_parse\_mode*
- is\_anonymous True, if the poll needs to be anonymous, defaults to True
- type Poll type, 'quiz' or 'regular', defaults to 'regular'
- allows\_multiple\_answers True, if the poll allows multiple answers, ignored for polls in quiz mode, defaults to False

- correct\_option\_id 0-based identifier of the correct answer option, required for polls in quiz mode
- **explanation** 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** Mode for parsing entities in the explanation. See formatting options for more details.
- **explanation\_entities** A JSON-serialized list of special entities that appear in the poll explanation. It can be specified instead of *explanation\_parse\_mode*
- **open\_period** Amount of time in seconds the poll will be active after creation, 5-600. Can't be used together with *close\_date*.
- **close\_date** 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** Pass True if the poll needs to be immediately closed. This can be useful for poll preview.
- disable\_notification Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- **message\_effect\_id** Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send\_poll.SendPoll

Shortcut for method <code>aiogram.methods.send\_dice.SendDice</code> will automatically fill method attributes:

chat\_id

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

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

# **Parameters**

- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- message\_thread\_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **emoji** 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 Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send\_dice.SendDice

Shortcut for method <code>aiogram.methods.send\_sticker.SendSticker</code> will automatically fill method attributes:

• chat\_id

Use this method to send static .WEBP, animated .TGS, or video .WEBM stickers. On success, the sent aiogram.types.message.Message is returned.

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

#### **Parameters**

- **sticker** Sticker to send. Pass a file\_id as String to send a file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a .WEBP sticker from the Internet, or upload a new .WEBP, .TGS, or .WEBM sticker using multipart/formdata. *More information on Sending Files* ». Video and animated stickers can't be sent via an HTTP URL.
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- **message\_thread\_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only

- **emoji** Emoji associated with the sticker; only for just uploaded stickers
- disable\_notification Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send\_sticker.SendSticker

answer\_venue(latitude: float, longitude: float, title: str, address: str, business\_connection\_id: Optional[str] = None, message\_thread\_id: Optional[int] = None, foursquare\_id: Optional[str] = None, foursquare\_type: Optional[str] = None, google\_place\_id: Optional[str] = None, google\_place\_type: Optional[str] = None, disable\_notification: Optional[bool] = None, protect\_content: Optional[Union[bool, Default]] = <Default('protect\_content')>, message\_effect\_id: Optional[str] = None, reply\_parameters: Optional[ReplyParameters] = None, reply\_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None, allow\_sending\_without\_reply: Optional[bool] = None, reply\_to\_message\_id: Optional[int] = None, \*\*kwargs: Any) → SendVenue

Shortcut for method <code>aiogram.methods.send\_venue.SendVenue</code> will automatically fill method attributes:

• chat\_id

Use this method to send information about a venue. On success, the sent aiogram.types.message. Message is returned.

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

## **Parameters**

- latitude Latitude of the venue
- longitude Longitude of the venue
- title Name of the venue
- address Address of the venue
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- **message\_thread\_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **foursquare\_id** Foursquare identifier of the venue
- **foursquare\_type** Foursquare type of the venue, if known. (For example, 'arts\_entertainment/default', 'arts\_entertainment/aquarium' or 'food/icecream'.)
- google\_place\_id Google Places identifier of the venue

- **google\_place\_type** Google Places type of the venue. (See supported types.)
- disable\_notification Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send\_venue.SendVenue

Shortcut for method aiogram.methods.send\_video.SendVideo will automatically fill method attributes:

• chat\_id

Use this method to send video files, Telegram clients support MPEG4 videos (other formats may be sent as aiogram.types.document.Document). On success, the sent aiogram.types.message.Message is returned. Bots can currently send video files of up to 50 MB in size, this limit may be changed in the future.

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

#### **Parameters**

- **video** Video to send. Pass a file\_id as String to send a video that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a video from the Internet, or upload a new video using multipart/form-data. *More information on Sending Files* »
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- **message\_thread\_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only

- **duration** Duration of sent video in seconds
- width Video width
- height Video height
- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file\_attach\_name>' if the thumbnail was uploaded using multipart/form-data under <file\_attach\_name>. *More information on Sending Files*
- **caption** Video caption (may also be used when resending videos by *file\_id*), 0-1024 characters after entities parsing
- parse\_mode Mode for parsing entities in the video caption. See formatting options for more details.
- **caption\_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse\_mode*
- **show\_caption\_above\_media** Pass True, if the caption must be shown above the message media
- has\_spoiler Pass True if the video needs to be covered with a spoiler animation
- **supports\_streaming** Pass True if the uploaded video is suitable for streaming
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send\_video.SendVideo

```
answer_video_note (video_note: Union[InputFile, str], business_connection_id: Optional[str] = None, message_thread_id: Optional[int] = None, duration: Optional[int] = None, length: Optional[int] = None, thumbnail: Optional[InputFile] = None, disable_notification: Optional[bool] = None, protect_content: Optional[Union[bool, Default]] = <Default('protect_content')>, message_effect_id: Optional[str] = None, reply_parameters: Optional[ReplyParameters] = None, reply_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None, allow_sending_without_reply: Optional[bool] = None, reply_to_message_id: Optional[int] = None, **kwargs: Any) → SendVideoNote
```

Shortcut for method <code>aiogram.methods.send\_video\_note.SendVideoNote</code> will automatically fill method attributes:

• chat\_id

As of v.4.0, Telegram clients support rounded square MPEG4 videos of up to 1 minute long. Use this method to send video messages. On success, the sent aiogram.types.message.Message is returned.

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

#### **Parameters**

- video\_note Video note to send. Pass a file\_id as String to send a video note that exists
  on the Telegram servers (recommended) or upload a new video using multipart/form-data.
   *More information on Sending Files* ». Sending video notes by a URL is currently unsupported
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- message\_thread\_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- duration Duration of sent video in seconds
- length Video width and height, i.e. diameter of the video message
- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file\_attach\_name>' if the thumbnail was uploaded using multipart/form-data under <file\_attach\_name>. *More information on Sending Files*""
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

#### Returns

instance of method aiogram.methods.send\_video\_note.SendVideoNote

Shortcut for method aiogram.methods.send\_voice.SendVoice will automatically fill method attributes:

### • chat\_id

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, or in .MP3 format, or in .M4A format (other formats may be sent as <code>aiogram.types.audio.Audio</code> or <code>aiogram.types.document.Document</code>). On success, the sent <code>aiogram.types.message.Message</code> is returned. Bots can currently send voice messages of up to 50 MB in size, this limit may be changed in the future.

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

#### **Parameters**

- **voice** Audio file to send. Pass a file\_id as String to send a file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a file from the Internet, or upload a new one using multipart/form-data. *More information on Sending Files* »
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- message\_thread\_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- caption Voice message caption, 0-1024 characters after entities parsing
- parse\_mode Mode for parsing entities in the voice message caption. See formatting options for more details.
- **caption\_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse\_mode*
- duration Duration of the voice message in seconds
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found

• reply\_to\_message\_id - If the message is a reply, ID of the original message

#### Returns

instance of method aiogram.methods.send\_voice.SendVoice

#### **ChatPermissions**

class aiogram.types.chat\_permissions.ChatPermissions(\*, can\_send\_messages: bool | None = None, can\_send\_audios: bool | None = None, can\_send\_documents: bool | None = None, can\_send\_photos: bool | None = None, can\_send\_videos: bool | None = None, can\_send\_video\_notes: bool | None = None, can\_send\_voice\_notes: bool | None = None, can\_send\_polls: bool | None = None, can\_send\_other\_messages: bool | None = None, can\_add\_web\_page\_previews: bool | None = None, can\_invite\_users: bool | None = None, can\_pin\_messages: bool | None = None, can\_manage\_topics: bool | None = None, \*\*extra\_data: Any)

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

Source: https://core.telegram.org/bots/api#chatpermissions

#### can\_send\_messages: bool | None

Optional. True, if the user is allowed to send text messages, contacts, giveaways, giveaway winners, invoices, locations and venues

# can\_send\_audios: bool | None

Optional. True, if the user is allowed to send audios

### can\_send\_documents: bool | None

Optional. True, if the user is allowed to send documents

## can\_send\_photos: bool | None

Optional. True, if the user is allowed to send photos

### can\_send\_videos: bool | None

Optional. True, if the user is allowed to send videos

## can\_send\_video\_notes: bool | None

Optional. True, if the user is allowed to send video notes

## can\_send\_voice\_notes: bool | None

Optional. True, if the user is allowed to send voice notes

### model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

# $model_post_init(context: Any, /) \rightarrow None$

We need to both initialize private attributes and call the user-defined model\_post\_init method.

## can\_send\_polls: bool | None

Optional. True, if the user is allowed to send polls

## can\_send\_other\_messages: bool | None

Optional. True, if the user is allowed to send animations, games, stickers and use inline bots

# can\_add\_web\_page\_previews: bool | None

Optional. True, if the user is allowed to add web page previews to their messages

# can\_change\_info: bool | None

*Optional*. True, if the user is allowed to change the chat title, photo and other settings. Ignored in public supergroups

## can\_invite\_users: bool | None

Optional. True, if the user is allowed to invite new users to the chat

# can\_pin\_messages: bool | None

Optional. True, if the user is allowed to pin messages. Ignored in public supergroups

## can\_manage\_topics: bool | None

Optional. True, if the user is allowed to create forum topics. If omitted defaults to the value of can\_pin\_messages

# **ChatPhoto**

This object represents a chat photo.

Source: https://core.telegram.org/bots/api#chatphoto

# small\_file\_id: str

File identifier of small (160x160) chat photo. This file\_id can be used only for photo download and only for as long as the photo is not changed.

# small\_file\_unique\_id: str

Unique file identifier of small (160x160) chat photo, which is supposed to be the same over time and for different bots. Can't be used to download or reuse the file.

# model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

# $model_post_init(context: Any, /) \rightarrow None$

We need to both initialize private attributes and call the user-defined model\_post\_init method.

# big\_file\_id: str

File identifier of big (640x640) chat photo. This file\_id can be used only for photo download and only for as long as the photo is not changed.

# big\_file\_unique\_id: str

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

# **ChatShared**

This object contains information about a chat that was shared with the bot using a aiogram.types. keyboard\_button\_request\_chat.KeyboardButtonRequestChat button.

Source: https://core.telegram.org/bots/api#chatshared

### request\_id: int

Identifier of the request

## chat\_id: int

Identifier of the shared chat. This number may have more than 32 significant bits and some programming languages may have difficulty/silent defects in interpreting it. But it has at most 52 significant bits, so a 64-bit integer or double-precision float type are safe for storing this identifier. The bot may not have access to the chat and could be unable to use this identifier, unless the chat is already known to the bot by some other means.

```
title: str | None
```

Optional. Title of the chat, if the title was requested by the bot.

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
username: str | None
```

Optional. Username of the chat, if the username was requested by the bot and available.

```
photo: List[PhotoSize] | None
```

Optional. Available sizes of the chat photo, if the photo was requested by the bot

# **Contact**

This object represents a phone contact.

Source: https://core.telegram.org/bots/api#contact

phone\_number: str

Contact's phone number

first\_name: str

Contact's first name

last\_name: str | None

Optional. Contact's last name

model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding  ${\it ComputedFieldInfo}$  objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
user_id: int | None
```

*Optional*. Contact's user identifier in Telegram. This number may have more than 32 significant bits and some programming languages may have difficulty/silent defects in interpreting it. But it has at most 52 significant bits, so a 64-bit integer or double-precision float type are safe for storing this identifier.

```
vcard: str | None
```

Optional. Additional data about the contact in the form of a vCard

### **Dice**

```
class aiogram.types.dice.Dice(*, emoji: str, value: int, **extra_data: Any)
```

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

Source: https://core.telegram.org/bots/api#dice

```
emoji: str
```

Emoji on which the dice throw animation is based

```
value: int
```

Value of the dice, 1-6 for ", " and " base emoji, 1-5 for " and " base emoji, 1-64 for " base emoji

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

class aiogram.types.dice.DiceEmoji

```
DICE = ''

DART = ''

BASKETBALL = ''

FOOTBALL = ''

SLOT_MACHINE = ''

BOWLING = ''
```

#### **Document**

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

Source: https://core.telegram.org/bots/api#document

# file\_id: str

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

## file\_unique\_id: str

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

#### thumbnail: PhotoSize | None

Optional. Document thumbnail as defined by the sender

# model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
file_name: str | None
```

Optional. Original filename as defined by the sender

```
mime_type: str | None
```

Optional. MIME type of the file as defined by the sender

## file\_size: int | None

*Optional.* File size in bytes. It can be bigger than 2^31 and some programming languages may have difficulty/silent defects in interpreting it. But it has at most 52 significant bits, so a signed 64-bit integer or double-precision float type are safe for storing this value.

# ExternalReplyInfo

class aiogram.types.external\_reply\_info.ExternalReplyInfo(\*, origin: MessageOriginUser |

MessageOriginHiddenUser | MessageOriginChat | MessageOriginChannel, chat: Chat | None = None, message\_id: int | None = *None*, *link\_preview\_options*:  $LinkPreviewOptions \mid None = None,$ animation: Animation | None = None, *audio:* Audio | None = None, document: Document | None = None, paid media: PaidMediaInfo | None = None, photo: *List*[PhotoSize] | *None* = *None*, *sticker*: Sticker | None = None, story: Story | None = None, video: Video | None = *None*, *video note*: VideoNote | *None* = *None*, *voice*: Voice | None = None, has media spoiler:  $bool \mid None = None$ , *contact:* Contact | *None* = *None*, *dice:* Dice | None = None, game: Game | None= None, giveaway: Giveaway | None = None, giveaway winners: GiveawayWinners | None = None, *invoice*: Invoice | None = None, *location:* Location | None = None, poll:Poll | None = None, venue: Venue | None

= None, \*\*extra data: Any)

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

Source: https://core.telegram.org/bots/api#externalreplyinfo origin: MessageOriginUser | MessageOriginHiddenUser | MessageOriginChat | MessageOriginChannel Origin of the message replied to by the given message chat: Chat | None Optional. Chat the original message belongs to. Available only if the chat is a supergroup or a channel. message\_id: int | None Optional. Unique message identifier inside the original chat. Available only if the original chat is a supergroup or a channel. link\_preview\_options: LinkPreviewOptions | None Optional. Options used for link preview generation for the original message, if it is a text message animation: Animation | None Optional. Message is an animation, information about the animation audio: Audio | None Optional. Message is an audio file, information about the file document: Document | None Optional. Message is a general file, information about the file paid\_media: PaidMediaInfo | None Optional. Message contains paid media; information about the paid media photo: List[PhotoSize] | None Optional. Message is a photo, available sizes of the photo sticker: Sticker | None Optional. Message is a sticker, information about the sticker story: Story | None Optional. Message is a forwarded story video: Video | None Optional. Message is a video, information about the video model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {} A dictionary of computed field names and their corresponding ComputedFieldInfo objects.  $model_post_init(context: Any, /) \rightarrow None$ We need to both initialize private attributes and call the user-defined model\_post\_init method. video\_note: VideoNote | None Optional. Message is a video note, information about the video message voice: Voice | None Optional. Message is a voice message, information about the file has\_media\_spoiler: bool | None Optional. True, if the message media is covered by a spoiler animation contact: Contact | None Optional. Message is a shared contact, information about the contact

dice: Dice | None

Optional. Message is a dice with random value

game: Game | None

Optional. Message is a game, information about the game. More about games »

giveaway: Giveaway | None

Optional. Message is a scheduled giveaway, information about the giveaway

giveaway\_winners: GiveawayWinners | None

Optional. A giveaway with public winners was completed

invoice: *Invoice* | None

Optional. Message is an invoice for a payment, information about the invoice. More about payments »

location: Location | None

Optional. Message is a shared location, information about the location

poll: Poll | None

Optional. Message is a native poll, information about the poll

venue: Venue | None

Optional. Message is a venue, information about the venue

### File

**class** aiogram.types.file.**File**(\*, file\_id: str, file\_unique\_id: str, file\_size: int | None = None, file\_path: str | None = None, \*\*extra\_data: Any)

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

The maximum file size to download is 20 MB

Source: https://core.telegram.org/bots/api#file

file\_id: str

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

file\_unique\_id: str

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

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

 $model_post_init(context: Any, /) \rightarrow None$ 

We need to both initialize private attributes and call the user-defined model\_post\_init method.

file\_size: int | None

*Optional.* File size in bytes. It can be bigger than 2^31 and some programming languages may have difficulty/silent defects in interpreting it. But it has at most 52 significant bits, so a signed 64-bit integer or double-precision float type are safe for storing this value.

file\_path: str | None

*Optional*. File path. Use https://api.telegram.org/file/bot<token>/<file\_path> to get the file.

# **ForceReply**

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. Not supported in channels and for messages sent on behalf of a Telegram Business account.

**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:

- Explain the user how to send a command with parameters (e.g. /newpoll question answer1 answer2). May be appealing for hardcore users but lacks modern day polish.
- Guide the user through a step-by-step process. 'Please send me your question', 'Cool, now let's add the first answer option', 'Great. Keep adding answer options, then send /done when you're ready'.

The last option is definitely more attractive. And if you use <code>aiogram.types.force\_reply.ForceReply</code> 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.

Source: https://core.telegram.org/bots/api#forcereply

# force\_reply: Literal[True]

Shows reply interface to the user, as if they manually selected the bot's message and tapped 'Reply'

```
input_field_placeholder: str | None
```

Optional. The placeholder to be shown in the input field when the reply is active; 1-64 characters

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model post init method.

```
selective: bool | None
```

*Optional*. Use this parameter if you want to force reply from specific users only. Targets: 1) users that are @mentioned in the *text* of the *aiogram.types.message.Message* object; 2) if the bot's message is a reply to a message in the same chat and forum topic, sender of the original message.

# **ForumTopic**

This object represents a forum topic.

Source: https://core.telegram.org/bots/api#forumtopic

## message\_thread\_id: int

Unique identifier of the forum topic

```
name: str
```

Name of the topic

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

icon\_color: int

Color of the topic icon in RGB format

icon\_custom\_emoji\_id: str | None

Optional. Unique identifier of the custom emoji shown as the topic icon

# **ForumTopicClosed**

```
class aiogram.types.forum_topic_closed.ForumTopicClosed(**extra_data: Any)
```

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

Source: https://core.telegram.org/bots/api#forumtopicclosed

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

# **ForumTopicCreated**

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

Source: https://core.telegram.org/bots/api#forumtopiccreated

name: str

Name of the topic

icon color: int

Color of the topic icon in RGB format

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
icon_custom_emoji_id: str | None
```

Optional. Unique identifier of the custom emoji shown as the topic icon

# **ForumTopicEdited**

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

Source: https://core.telegram.org/bots/api#forumtopicedited

name: str | None

Optional. New name of the topic, if it was edited

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
icon_custom_emoji_id: str | None
```

Optional. New identifier of the custom emoji shown as the topic icon, if it was edited; an empty string if the icon was removed

# **ForumTopicReopened**

```
class aiogram.types.forum_topic_reopened.ForumTopicReopened(**extra_data: Any)
```

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

Source: https://core.telegram.org/bots/api#forumtopicreopened

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

# GeneralForumTopicHidden

```
class aiogram.types.general_forum_topic_hidden.GeneralForumTopicHidden(**extra_data: Any)
```

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

Source: https://core.telegram.org/bots/api#generalforumtopichidden

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

# GeneralForumTopicUnhidden

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

Source: https://core.telegram.org/bots/api#generalforumtopicunhidden

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

# Giveaway

This object represents a message about a scheduled giveaway.

Source: https://core.telegram.org/bots/api#giveaway

```
chats: List[Chat]
```

The list of chats which the user must join to participate in the giveaway

```
winners_selection_date: DateTime
```

Point in time (Unix timestamp) when winners of the giveaway will be selected

```
winner count: int
```

The number of users which are supposed to be selected as winners of the giveaway

```
only_new_members: bool | None
```

Optional. True, if only users who join the chats after the giveaway started should be eligible to win

```
has_public_winners: bool | None
```

Optional. True, if the list of giveaway winners will be visible to everyone

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
prize_description: str | None
```

Optional. Description of additional giveaway prize

```
country_codes: List[str] | None
```

*Optional*. A list of two-letter ISO 3166-1 alpha-2 country codes indicating the countries from which eligible users for the giveaway must come. If empty, then all users can participate in the giveaway. Users with a phone number that was bought on Fragment can always participate in giveaways.

```
prize_star_count: int | None
```

*Optional*. The number of Telegram Stars to be split between giveaway winners; for Telegram Star giveaways only

```
premium_subscription_month_count: int | None
```

*Optional*. The number of months the Telegram Premium subscription won from the giveaway will be active for; for Telegram Premium giveaways only

# GiveawayCompleted

class aiogram.types.giveaway\_completed.GiveawayCompleted(\*, winner\_count: int,

unclaimed\_prize\_count: int | None = None, giveaway\_message: Message | None = None, is\_star\_giveaway: bool | None = None, \*\*extra\_data: Any)

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

Source: https://core.telegram.org/bots/api#giveawaycompleted

winner\_count: int

Number of winners in the giveaway

unclaimed\_prize\_count: int | None

Optional. Number of undistributed prizes

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
giveaway_message: Message | None
```

Optional. Message with the giveaway that was completed, if it wasn't deleted

```
is_star_giveaway: bool | None
```

*Optional*. True, if the giveaway is a Telegram Star giveaway. Otherwise, currently, the giveaway is a Telegram Premium giveaway.

# **GiveawayCreated**

This object represents a service message about the creation of a scheduled giveaway.

Source: https://core.telegram.org/bots/api#giveawaycreated

```
prize_star_count: int | None
```

*Optional*. The number of Telegram Stars to be split between giveaway winners; for Telegram Star giveaways only

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

# **GiveawayWinners**

```
class aiogram.types.giveaway_winners.GiveawayWinners(*, chat: Chat, giveaway_message_id: int,
                                                                winners_selection_date: _datetime_serializer,
                                                                return_type=int, when_used=unless - none)],
                                                                winner_count: int, winners: List[User],
                                                                additional\_chat\_count: int \mid None = None,
                                                                prize star count: int | None = None,
                                                                premium subscription month count: int |
                                                                None = None, unclaimed_prize_count: int |
                                                                None = None, only\_new\_members: bool | None
                                                                = None, was_refunded: bool | None = None,
                                                                prize\_description: str \mid None = None,
                                                                **extra data: Any)
     This object represents a message about the completion of a giveaway with public winners.
     Source: https://core.telegram.org/bots/api#giveawaywinners
     chat: Chat
          The chat that created the giveaway
     giveaway_message_id: int
          Identifier of the message with the giveaway in the chat
     winners_selection_date: DateTime
          Point in time (Unix timestamp) when winners of the giveaway were selected
     winner_count: int
          Total number of winners in the giveaway
     winners: List[User]
          List of up to 100 winners of the giveaway
     additional_chat_count: int | None
          Optional. The number of other chats the user had to join in order to be eligible for the giveaway
     model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
          A dictionary of computed field names and their corresponding ComputedFieldInfo objects.
     model_post_init(context: Any, /) \rightarrow None
          We need to both initialize private attributes and call the user-defined model_post_init method.
     prize_star_count: int | None
          Optional. The number of Telegram Stars that were split between giveaway winners; for Telegram Star
          giveaways only
     premium_subscription_month_count: int | None
          Optional. The number of months the Telegram Premium subscription won from the giveaway will be active
          for; for Telegram Premium giveaways only
     unclaimed_prize_count: int | None
          Optional. Number of undistributed prizes
     only_new_members: bool | None
          Optional. True, if only users who had joined the chats after the giveaway started were eligible to win
```

# was\_refunded: bool | None

Optional. True, if the giveaway was canceled because the payment for it was refunded

## prize\_description: str | None

Optional. Description of additional giveaway prize

## InaccessibleMessage

class aiogram.types.inaccessible\_message.InaccessibleMessage(\*, chat: Chat, message\_id: int, date: Literal[0] = 0, \*\*extra\_data: Any)

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

Source: https://core.telegram.org/bots/api#inaccessiblemessage

#### chat: Chat

Chat the message belonged to

## message\_id: int

Unique message identifier inside the chat

# date: Literal[0]

Always 0. The field can be used to differentiate regular and inaccessible messages.

```
answer(text: str, business_connection_id: Optional[str] = None, message_thread_id: Optional[int] = None, parse_mode: Optional[Union[str, Default]] = <Default('parse_mode')>, entities:

Optional[List[MessageEntity]] = None, link_preview_options: Optional[Union[LinkPreviewOptions, Default]] = <Default('link_preview')>, disable_notification: Optional[bool] = None, protect_content: Optional[Union[bool, Default]] = <Default('protect_content')>, message_effect_id: Optional[str] = None, reply_parameters: Optional[ReplyParameters] = None, reply_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None, allow_sending_without_reply: Optional[bool] = None, disable_web_page_preview: Optional[Union[bool, Default]] = <Default('link_preview_is_disabled')>, reply_to_message_id: Optional[int] = None, **kwargs: Any) → SendMessage
```

Shortcut for method  $aiogram.methods.send\_message.SendMessage$  will automatically fill method attributes:

## • chat\_id

Use this method to send text messages. On success, the sent aiogram.types.message.Message is returned

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

## **Parameters**

- text Text of the message to be sent, 1-4096 characters after entities parsing
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- **message\_thread\_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- parse\_mode Mode for parsing entities in the message text. See formatting options for more details.
- **entities** A JSON-serialized list of special entities that appear in message text, which can be specified instead of *parse\_mode*

- link\_preview\_options Link preview generation options for the message
- disable\_notification Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- disable\_web\_page\_preview Disables link previews for links in this message
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send\_message.SendMessage

reply(text: str, business\_connection\_id: Optional[str] = None, message\_thread\_id: Optional[int] = None, parse\_mode: Optional[Union[str, Default]] = <Default('parse\_mode')>, entities: Optional[List[MessageEntity]] = None, link\_preview\_options: Optional[Union[LinkPreviewOptions, Default]] = <Default('link\_preview')>, disable\_notification: Optional[bool] = None, protect\_content: Optional[Union[bool, Default]] = <Default('protect\_content')>, message\_effect\_id: Optional[str] = None, reply\_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None, allow\_sending\_without\_reply: Optional[bool] = None, disable\_web\_page\_preview: Optional[Union[bool, Default]] = <Default('link\_preview\_is\_disabled')>, \*\*kwargs: Any) → SendMessage

Shortcut for method  $aiogram.methods.send\_message.SendMessage$  will automatically fill method attributes:

- chat\_id
- reply\_parameters

Use this method to send text messages. On success, the sent aiogram.types.message.Message is returned.

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

- text Text of the message to be sent, 1-4096 characters after entities parsing
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- **message\_thread\_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- parse\_mode Mode for parsing entities in the message text. See formatting options for more details.
- **entities** A JSON-serialized list of special entities that appear in message text, which can be specified instead of *parse\_mode*
- link\_preview\_options Link preview generation options for the message

- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- disable\_web\_page\_preview Disables link previews for links in this message

instance of method aiogram.methods.send\_message.SendMessage

```
answer_animation(animation: Union[InputFile, str], business_connection_id: Optional[str] = None, message_thread_id: Optional[int] = None, duration: Optional[int] = None, width:

Optional[int] = None, height: Optional[int] = None, thumbnail: Optional[InputFile] = None, caption: Optional[str] = None, parse_mode: Optional[Union[str, Default]] = < Default('parse_mode')>, caption_entities: Optional[List[MessageEntity]] = None, show_caption_above_media: Optional[Union[bool, Default]] = < Default('show_caption_above_media')>, has_spoiler: Optional[bool] = None, disable_notification: Optional[bool] = None, protect_content: Optional[Union[bool, Default]] = < Default('protect_content')>, message_effect_id: Optional[str] = None, reply_parameters: Optional[ReplyParameters] = None, reply_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None, allow_sending_without_reply: Optional[bool] = None, reply_to_message_id: Optional[int] = None, **kwargs: Any) → SendAnimation
```

Shortcut for method  $aiogram.methods.send\_animation.SendAnimation$  will automatically fill method attributes:

• chat\_id

Use this method to send animation files (GIF or H.264/MPEG-4 AVC video without sound). On success, the sent aiogram.types.message.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 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. *More information on Sending Files* »
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- message\_thread\_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- duration Duration of sent animation in seconds
- width Animation width

- **height** Animation height
- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file\_attach\_name>' if the thumbnail was uploaded using multipart/form-data under <file\_attach\_name>. *More information on Sending Files*"""
- **caption** Animation caption (may also be used when resending animation by *file\_id*), 0-1024 characters after entities parsing
- parse\_mode Mode for parsing entities in the animation caption. See formatting options for more details.
- **caption\_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse\_mode*
- **show\_caption\_above\_media** Pass True, if the caption must be shown above the message media
- has\_spoiler Pass True if the animation needs to be covered with a spoiler animation
- disable\_notification Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send\_animation.SendAnimation

reply\_animation(animation: Union[InputFile, str], business\_connection\_id: Optional[str] = None, message\_thread\_id: Optional[int] = None, duration: Optional[int] = None, width:

Optional[int] = None, height: Optional[int] = None, thumbnail: Optional[InputFile] = None, caption: Optional[str] = None, parse\_mode: Optional[Union[str, Default]] = <Default('parse\_mode')>, caption\_entities: Optional[List[MessageEntity]] = None, show\_caption\_above\_media: Optional[Union[bool, Default]] = <Default('show\_caption\_above\_media')>, has\_spoiler: Optional[bool] = None, disable\_notification: Optional[bool] = None, protect\_content: Optional[Union[bool, Default]] = <Default('protect\_content')>, message\_effect\_id: Optional[str] = None, reply\_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None, allow\_sending\_without\_reply: Optional[bool] = None, \*\*kwargs: Any) → SendAnimation

Shortcut for method aiogram.methods.send\_animation.SendAnimation will automatically fill method attributes:

• chat\_id

• reply\_parameters

Use this method to send animation files (GIF or H.264/MPEG-4 AVC video without sound). On success, the sent *aiogram.types.message*. 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 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. More information on Sending Files »
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- **message\_thread\_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- duration Duration of sent animation in seconds
- width Animation width
- height Animation height
- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file\_attach\_name>' if the thumbnail was uploaded using multipart/form-data under <file\_attach\_name>. *More information on Sending Files*
- **caption** Animation caption (may also be used when resending animation by *file\_id*), 0-1024 characters after entities parsing
- parse\_mode Mode for parsing entities in the animation caption. See formatting options for more details.
- **caption\_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse\_mode*
- **show\_caption\_above\_media** Pass True, if the caption must be shown above the message media
- has\_spoiler Pass True if the animation needs to be covered with a spoiler animation
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found

instance of method aiogram.methods.send\_animation.SendAnimation

Shortcut for method aiogram.methods.send\_audio.SendAudio will automatically fill method attributes:

## • chat\_id

Use this method to send audio files, if you want Telegram clients to display them in the music player. Your audio must be in the .MP3 or .M4A format. On success, the sent <code>aiogram.types.message.Message</code> is returned. Bots can currently send audio files of up to 50 MB in size, this limit may be changed in the future. For sending voice messages, use the <code>aiogram.methods.send\_voice.SendVoice</code> method instead.

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

#### **Parameters**

- audio Audio file to send. Pass a file\_id as String to send an audio file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get an audio file from the Internet, or upload a new one using multipart/form-data. *More information on Sending Files* »
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- **message\_thread\_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- caption Audio caption, 0-1024 characters after entities parsing
- parse\_mode Mode for parsing entities in the audio caption. See formatting options for more details.
- **caption\_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse\_mode*
- duration Duration of the audio in seconds
- **performer** Performer
- title Track name
- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file\_attach\_name>' if the thumbnail was uploaded using multipart/form-data under <file\_attach\_name>. *More information on Sending Files*

X

- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send\_audio.SendAudio

Shortcut for method aiogram.methods.send\_audio.SendAudio will automatically fill method attributes:

- chat\_id
- reply\_parameters

Use this method to send audio files, if you want Telegram clients to display them in the music player. Your audio must be in the .MP3 or .M4A format. On success, the sent <code>aiogram.types.message.Message</code> is returned. Bots can currently send audio files of up to 50 MB in size, this limit may be changed in the future. For sending voice messages, use the <code>aiogram.methods.send\_voice.SendVoice</code> method instead.

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

## **Parameters**

- audio Audio file to send. Pass a file\_id as String to send an audio file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get an audio file from the Internet, or upload a new one using multipart/form-data. *More information on Sending Files* »
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- message\_thread\_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- caption Audio caption, 0-1024 characters after entities parsing
- parse\_mode Mode for parsing entities in the audio caption. See formatting options for more details.

- **caption\_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse\_mode*
- duration Duration of the audio in seconds
- **performer** Performer
- title Track name
- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file\_attach\_name>' if the thumbnail was uploaded using multipart/form-data under <file\_attach\_name>. *More information on Sending Files*\*\*
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found

instance of method aiogram.methods.send\_audio.SendAudio

answer\_contact(phone\_number: str, first\_name: str, business\_connection\_id: Optional[str] = None, message\_thread\_id: Optional[int] = None, last\_name: Optional[str] = None, vcard: Optional[str] = None, disable\_notification: Optional[bool] = None, protect\_content: Optional[Union[bool, Default]] = <Default('protect\_content')>, message\_effect\_id: Optional[str] = None, reply\_parameters: Optional[ReplyParameters] = None, reply\_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None, allow\_sending\_without\_reply: Optional[bool] = None, reply\_to\_message\_id: Optional[int] = None, \*\*kwargs: Any) → SendContact

Shortcut for method <code>aiogram.methods.send\_contact.SendContact</code> will automatically fill method attributes:

• chat\_id

Use this method to send phone contacts. On success, the sent aiogram.types.message.Message is returned.

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

- **phone\_number** Contact's phone number
- **first\_name** Contact's first name
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent

- message\_thread\_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- last\_name Contact's last name
- vcard Additional data about the contact in the form of a vCard, 0-2048 bytes
- disable\_notification Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send\_contact.SendContact

```
reply_contact(phone_number: str, first_name: str, business_connection_id: Optional[str] = None, message_thread_id: Optional[int] = None, last_name: Optional[str] = None, vcard: Optional[str] = None, disable_notification: Optional[bool] = None, protect_content: Optional[Union[bool, Default]] = <Default('protect_content')>, message_effect_id: Optional[str] = None, reply_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None, allow_sending_without_reply: Optional[bool] = None, **kwargs: Any) → SendContact
```

Shortcut for method  $aiogram.methods.send\_contact.SendContact$  will automatically fill method attributes:

- chat\_id
- reply\_parameters

Use this method to send phone contacts. On success, the sent aiogram.types.message.Message is returned.

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

#### **Parameters**

- phone\_number Contact's phone number
- **first\_name** Contact's first name
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- **message\_thread\_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- last\_name Contact's last name
- vcard Additional data about the contact in the form of a vCard, 0-2048 bytes

- disable\_notification Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found

instance of method aiogram.methods.send\_contact.SendContact

Shortcut for method <code>aiogram.methods.send\_document.SendDocument</code> will automatically fill method attributes:

• chat\_id

Use this method to send general files. On success, the sent aiogram.types.message.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** File to send. Pass a file\_id as String to send a file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a file from the Internet, or upload a new one using multipart/form-data. *More information on Sending Files* »
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- **message\_thread\_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file\_attach\_name>' if the thumbnail was uploaded using multipart/form-data under <file\_attach\_name>. *More information on Sending Files*

>>

- **caption** Document caption (may also be used when resending documents by *file\_id*), 0-1024 characters after entities parsing
- parse\_mode Mode for parsing entities in the document caption. See formatting options for more details.
- **caption\_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse\_mode*
- **disable\_content\_type\_detection** Disables automatic server-side content type detection for files uploaded using multipart/form-data
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_parameters** Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send\_document.SendDocument

Shortcut for method <code>aiogram.methods.send\_document.SendDocument</code> will automatically fill method attributes:

- chat\_id
- reply\_parameters

Use this method to send general files. On success, the sent aiogram.types.message.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** – File to send. Pass a file\_id as String to send a file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a file from the Internet, or upload a new one using multipart/form-data. *More information on Sending Files* »

- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- **message\_thread\_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file\_attach\_name>' if the thumbnail was uploaded using multipart/form-data under <file\_attach\_name>. *More information on Sending Files*
- **caption** Document caption (may also be used when resending documents by *file\_id*), 0-1024 characters after entities parsing
- parse\_mode Mode for parsing entities in the document caption. See formatting options for more details.
- **caption\_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse\_mode*
- **disable\_content\_type\_detection** Disables automatic server-side content type detection for files uploaded using multipart/form-data
- disable\_notification Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found

instance of method aiogram.methods.send\_document.SendDocument

 $\begin{tabular}{ll} \textbf{answer\_game} (game\_short\_name: str, business\_connection\_id: Optional[str] = None, message\_thread\_id: \\ Optional[int] = None, disable\_notification: Optional[bool] = None, protect\_content: \\ Optional[Union[bool, Default]] = & Optional[violetot] & Optional[str] = None, reply\_parameters: Optional[ReplyParameters] = None, reply\_markup: \\ Optional[InlineKeyboardMarkup] = None, allow\_sending\_without\_reply: Optional[bool] = \\ None, reply\_to\_message\_id: Optional[int] = None, **kwargs: Any) & SendGame \\ \end{tabular}$ 

Shortcut for method aiogram.methods.send\_game.SendGame will automatically fill method attributes:

• chat id

Use this method to send a game. On success, the sent aiogram.types.message.Message is returned.

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

## **Parameters**

• game\_short\_name — Short name of the game, serves as the unique identifier for the game. Set up your games via @BotFather.

- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- message\_thread\_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_parameters** Description of the message to reply to
- **reply\_markup** 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.
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send\_game.SendGame

 $\textbf{reply\_game}(game\_short\_name: str, business\_connection\_id: Optional[str] = None, message\_thread\_id: \\ Optional[int] = None, disable\_notification: Optional[bool] = None, protect\_content: \\ Optional[Union[bool, Default]] = <Default('protect\_content')>, message\_effect\_id: \\ Optional[str] = None, reply\_markup: Optional[InlineKeyboardMarkup] = None, \\ allow\_sending\_without\_reply: Optional[bool] = None, **kwargs: Any) \rightarrow SendGame$ 

 $Shortcut\ for\ method\ aiogram.methods.send\_game.SendGame\ will\ automatically\ fill\ method\ attributes:$ 

- chat\_id
- reply\_parameters

Use this method to send a game. On success, the sent aiogram.types.message.Message is returned.

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

# **Parameters**

- game\_short\_name Short name of the game, serves as the unique identifier for the game. Set up your games via @BotFather.
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- message\_thread\_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_markup** 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.

• allow\_sending\_without\_reply - Pass True if the message should be sent even if the specified replied-to message is not found

#### Returns

instance of method aiogram.methods.send\_game.SendGame

Shortcut for method aiogram.methods.send\_invoice.SendInvoice will automatically fill method attributes:

• chat\_id

Use this method to send invoices. On success, the sent aiogram.types.message.Message is returned.

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

- title Product name, 1-32 characters
- **description** Product description, 1-255 characters
- **payload** Bot-defined invoice payload, 1-128 bytes. This will not be displayed to the user, use it for your internal processes.
- **currency** Three-letter ISO 4217 currency code, see more on currencies. Pass 'XTR' for payments in Telegram Stars.
- prices Price breakdown, a JSON-serialized list of components (e.g. product price, tax, discount, delivery cost, delivery tax, bonus, etc.). Must contain exactly one item for payments in Telegram Stars.
- **message\_thread\_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **provider\_token** Payment provider token, obtained via @BotFather. Pass an empty string for payments in Telegram Stars.
- max\_tip\_amount The maximum accepted amount for tips in the *smallest units* of the currency (integer, **not** float/double). For example, for a maximum tip of US\$ 1.45 pass max\_tip\_amount = 145. See the *exp* parameter in currencies.json, it shows the number of digits past the decimal point for each currency (2 for the majority of currencies). Defaults to 0. Not supported for payments in Telegram Stars.
- **suggested\_tip\_amounts** A JSON-serialized array of suggested amounts of tips in the *smallest units* of the currency (integer, **not** float/double). At most 4 suggested tip amounts

- can be specified. The suggested tip amounts must be positive, passed in a strictly increased order and must not exceed *max tip amount*.
- **start\_parameter** Unique deep-linking parameter. If left empty, **forwarded copies** of the sent message will have a *Pay* button, allowing multiple users to pay directly from the forwarded message, using the same invoice. If non-empty, forwarded copies of the sent message will have a *URL* button with a deep link to the bot (instead of a *Pay* button), with the value used as the start parameter
- provider\_data JSON-serialized data about the invoice, which will be shared with the
  payment provider. A detailed description of required fields should be provided by the payment provider.
- **photo\_url** URL of the product photo for the invoice. Can be a photo of the goods or a marketing image for a service. People like it better when they see what they are paying for.
- **photo\_size** Photo size in bytes
- photo\_width Photo width
- photo\_height Photo height
- **need\_name** Pass True if you require the user's full name to complete the order. Ignored for payments in Telegram Stars.
- **need\_phone\_number** Pass True if you require the user's phone number to complete the order. Ignored for payments in Telegram Stars.
- need\_email Pass True if you require the user's email address to complete the order. Ignored for payments in Telegram Stars.
- **need\_shipping\_address** Pass True if you require the user's shipping address to complete the order. Ignored for payments in Telegram Stars.
- **send\_phone\_number\_to\_provider** Pass True if the user's phone number should be sent to the provider. Ignored for payments in Telegram Stars.
- **send\_email\_to\_provider** Pass True if the user's email address should be sent to the provider. Ignored for payments in Telegram Stars.
- **is\_flexible** Pass True if the final price depends on the shipping method. Ignored for payments in Telegram Stars.
- disable\_notification Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_parameters** Description of the message to reply to
- **reply\_markup** 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.
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send\_invoice.SendInvoice

Shortcut for method <code>aiogram.methods.send\_invoice.SendInvoice</code> will automatically fill method attributes:

- chat\_id
- reply\_parameters

Use this method to send invoices. On success, the sent aiogram.types.message.Message is returned.

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

- title Product name, 1-32 characters
- **description** Product description, 1-255 characters
- payload Bot-defined invoice payload, 1-128 bytes. This will not be displayed to the user, use it for your internal processes.
- **currency** Three-letter ISO 4217 currency code, see more on currencies. Pass 'XTR' for payments in Telegram Stars.
- **prices** Price breakdown, a JSON-serialized list of components (e.g. product price, tax, discount, delivery cost, delivery tax, bonus, etc.). Must contain exactly one item for payments in Telegram Stars.
- message\_thread\_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **provider\_token** Payment provider token, obtained via @BotFather. Pass an empty string for payments in Telegram Stars.
- max\_tip\_amount The maximum accepted amount for tips in the *smallest units* of the currency (integer, **not** float/double). For example, for a maximum tip of US\$ 1.45 pass max\_tip\_amount = 145. See the *exp* parameter in currencies.json, it shows the number of digits past the decimal point for each currency (2 for the majority of currencies). Defaults to 0. Not supported for payments in Telegram Stars.
- **suggested\_tip\_amounts** A JSON-serialized array of suggested amounts of tips in the *smallest units* of the currency (integer, **not** float/double). At most 4 suggested tip amounts can be specified. The suggested tip amounts must be positive, passed in a strictly increased order and must not exceed *max\_tip\_amount*.
- **start\_parameter** Unique deep-linking parameter. If left empty, **forwarded copies** of the sent message will have a *Pay* button, allowing multiple users to pay directly from the forwarded message, using the same invoice. If non-empty, forwarded copies of the sent

message will have a *URL* button with a deep link to the bot (instead of a *Pay* button), with the value used as the start parameter

- provider\_data JSON-serialized data about the invoice, which will be shared with the
  payment provider. A detailed description of required fields should be provided by the payment provider.
- **photo\_url** URL of the product photo for the invoice. Can be a photo of the goods or a marketing image for a service. People like it better when they see what they are paying for.
- **photo\_size** Photo size in bytes
- photo\_width Photo width
- photo\_height Photo height
- **need\_name** Pass True if you require the user's full name to complete the order. Ignored for payments in Telegram Stars.
- **need\_phone\_number** Pass True if you require the user's phone number to complete the order. Ignored for payments in Telegram Stars.
- need\_email Pass True if you require the user's email address to complete the order. Ignored for payments in Telegram Stars.
- **need\_shipping\_address** Pass True if you require the user's shipping address to complete the order. Ignored for payments in Telegram Stars.
- **send\_phone\_number\_to\_provider** Pass True if the user's phone number should be sent to the provider. Ignored for payments in Telegram Stars.
- **send\_email\_to\_provider** Pass True if the user's email address should be sent to the provider. Ignored for payments in Telegram Stars.
- is\_flexible Pass True if the final price depends on the shipping method. Ignored for payments in Telegram Stars.
- disable\_notification Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_markup** 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.
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found

# Returns

instance of method aiogram.methods.send\_invoice.SendInvoice

Shortcut for method <code>aiogram.methods.send\_location</code>. <code>SendLocation</code> will automatically fill method attributes:

• chat\_id

Use this method to send point on the map. On success, the sent aiogram.types.message.Message is returned.

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

#### **Parameters**

- latitude Latitude of the location
- longitude Longitude of the location
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- message\_thread\_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- horizontal\_accuracy The radius of uncertainty for the location, measured in meters;
   0-1500
- **live\_period** Period in seconds during which the location will be updated (see Live Locations, should be between 60 and 86400, or 0x7FFFFFFF for live locations that can be edited indefinitely.
- **heading** For live locations, a direction in which the user is moving, in degrees. Must be between 1 and 360 if specified.
- **proximity\_alert\_radius** 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** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

#### Returns

instance of method aiogram.methods.send\_location.SendLocation

Shortcut for method <code>aiogram.methods.send\_location</code>. SendLocation will automatically fill method attributes:

- · chat id
- reply\_parameters

Use this method to send point on the map. On success, the sent aiogram.types.message.Message is returned.

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

## **Parameters**

- latitude Latitude of the location
- longitude Longitude of the location
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- **message\_thread\_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **horizontal\_accuracy** The radius of uncertainty for the location, measured in meters; 0-1500
- **live\_period** Period in seconds during which the location will be updated (see Live Locations, should be between 60 and 86400, or 0x7FFFFFFF for live locations that can be edited indefinitely.
- **heading** For live locations, a direction in which the user is moving, in degrees. Must be between 1 and 360 if specified.
- **proximity\_alert\_radius** 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** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found

instance of method aiogram.methods.send\_location.SendLocation

Shortcut for method <code>aiogram.methods.send\_media\_group.SendMediaGroup</code> will automatically fill method attributes:

• chat\_id

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

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

#### **Parameters**

- media A JSON-serialized array describing messages to be sent, must include 2-10 items
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- **message\_thread\_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- disable\_notification Sends messages silently. Users will receive a notification with no sound.
- **protect\_content** Protects the contents of the sent messages from forwarding and saving
- **message\_effect\_id** Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the messages are a reply, ID of the original message

## **Returns**

instance of method aiogram.methods.send\_media\_group.SendMediaGroup

Shortcut for method aiogram.methods.send\_media\_group.SendMediaGroup will automatically fill method attributes:

• chat\_id

• reply\_parameters

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

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

#### **Parameters**

- media A JSON-serialized array describing messages to be sent, must include 2-10 items
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- **message\_thread\_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- disable\_notification Sends messages silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent messages from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found

## Returns

instance of method aiogram.methods.send\_media\_group.SendMediaGroup

Shortcut for method <code>aiogram.methods.send\_photo.SendPhoto</code> will automatically fill method attributes:

• chat\_id

Use this method to send photos. On success, the sent aiogram. types.message.Message is returned.

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

## **Parameters**

- **photo** Photo to send. Pass a file\_id as String to send a photo that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a photo from the Internet, or upload a new photo using multipart/form-data. The photo must be at most 10 MB in size. The photo's width and height must not exceed 10000 in total. Width and height ratio must be at most 20. *More information on Sending Files* »
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent

- message\_thread\_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **caption** Photo caption (may also be used when resending photos by *file\_id*), 0-1024 characters after entities parsing
- parse\_mode Mode for parsing entities in the photo caption. See formatting options for more details.
- **caption\_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse\_mode*
- **show\_caption\_above\_media** Pass True, if the caption must be shown above the message media
- has\_spoiler Pass True if the photo needs to be covered with a spoiler animation
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send\_photo.SendPhoto

Shortcut for method aiogram.methods.send\_photo.SendPhoto will automatically fill method attributes:

- chat\_id
- reply\_parameters

Use this method to send photos. On success, the sent aiogram. types.message.Message is returned.

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

- **photo** Photo to send. Pass a file\_id as String to send a photo that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a photo from the Internet, or upload a new photo using multipart/form-data. The photo must be at most 10 MB in size. The photo's width and height must not exceed 10000 in total. Width and height ratio must be at most 20. *More information on Sending Files* »
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- message\_thread\_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **caption** Photo caption (may also be used when resending photos by *file\_id*), 0-1024 characters after entities parsing
- parse\_mode Mode for parsing entities in the photo caption. See formatting options for more details.
- **caption\_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse\_mode*
- **show\_caption\_above\_media** Pass True, if the caption must be shown above the message media
- has\_spoiler Pass True if the photo needs to be covered with a spoiler animation
- disable\_notification Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found

instance of method aiogram.methods.send\_photo.SendPhoto

answer\_poll(question: str, options: List[Union[InputPollOption, str]], business\_connection\_id:

Optional[str] = None, message\_thread\_id: Optional[int] = None, question\_parse\_mode:

Optional[Union[str, Default]] = <Default('parse\_mode')>, question\_entities:

Optional[List[MessageEntity]] = None, is\_anonymous: Optional[bool] = None, type:

Optional[str] = None, allows\_multiple\_answers: Optional[bool] = None, correct\_option\_id:

Optional[int] = None, explanation: Optional[str] = None, explanation\_parse\_mode:

Optional[Union[str, Default]] = <Default('parse\_mode')>, explanation\_entities:

Optional[List[MessageEntity]] = None, open\_period: Optional[int] = None, close\_date:

Optional[Union[datetime.datetime, datetime.timedelta, int]] = None, is\_closed:

Optional[bool] = None, disable\_notification: Optional[bool] = None, protect\_content:

Optional[Union[bool, Default]] = <Default('protect\_content')>, message\_effect\_id:

Optional[str] = None, reply\_parameters: Optional[ReplyParameters] = None, reply\_markup:

Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove,
ForceReply]] = None, allow\_sending\_without\_reply: Optional[bool] = None,
reply to message id: Optional[int] = None, \*\*kwargs: Any) → SendPoll

Shortcut for method aiogram.methods.send\_poll.SendPoll will automatically fill method attributes:

• chat\_id

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

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

- question Poll question, 1-300 characters
- options A JSON-serialized list of 2-10 answer options
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- message\_thread\_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **question\_parse\_mode** Mode for parsing entities in the question. See formatting options for more details. Currently, only custom emoji entities are allowed
- **question\_entities** A JSON-serialized list of special entities that appear in the poll question. It can be specified instead of *question\_parse\_mode*
- is\_anonymous True, if the poll needs to be anonymous, defaults to True
- **type** Poll type, 'quiz' or 'regular', defaults to 'regular'
- allows\_multiple\_answers True, if the poll allows multiple answers, ignored for polls in quiz mode, defaults to False
- **correct\_option\_id** 0-based identifier of the correct answer option, required for polls in quiz mode
- **explanation** 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** Mode for parsing entities in the explanation. See formatting options for more details.
- **explanation\_entities** A JSON-serialized list of special entities that appear in the poll explanation. It can be specified instead of *explanation\_parse\_mode*
- **open\_period** Amount of time in seconds the poll will be active after creation, 5-600. Can't be used together with *close\_date*.
- **close\_date** 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 Pass True if the poll needs to be immediately closed. This can be useful for poll preview.
- disable\_notification Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to

- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send\_poll.SendPoll

reply\_poll (question: str, options: List[Union[InputPollOption, str]], business\_connection\_id: Optional[str] = None, message\_thread\_id: Optional[int] = None, question\_parse\_mode: Optional[Union[str, Default]] = <Default('parse\_mode')>, question\_entities: Optional[List[MessageEntity]] = None, is\_anonymous: Optional[bool] = None, type: Optional[str] = None, allows\_multiple\_answers: Optional[bool] = None, correct\_option\_id: Optional[int] = None, explanation: Optional[str] = None, explanation\_parse\_mode: Optional[Union[str, Default]] = <Default('parse\_mode')>, explanation\_entities: Optional[List[MessageEntity]] = None, open\_period: Optional[int] = None, close\_date: Optional[Union[datetime.datetime, datetime.timedelta, int]] = None, is\_closed: Optional[bool] = None, disable\_notification: Optional[bool] = None, protect\_content: Optional[Union[bool, Default]] = <Default('protect\_content')>, message\_effect\_id: Optional[str] = None, reply\_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None, allow\_sending\_without\_reply: Optional[bool] = None, \*\*kwargs: Any) → SendPoll

Shortcut for method aiogram.methods.send\_poll.SendPoll will automatically fill method attributes:

- · chat id
- reply\_parameters

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

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

#### **Parameters**

- question Poll question, 1-300 characters
- options A JSON-serialized list of 2-10 answer options
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- **message\_thread\_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **question\_parse\_mode** Mode for parsing entities in the question. See formatting options for more details. Currently, only custom emoji entities are allowed
- **question\_entities** A JSON-serialized list of special entities that appear in the poll question. It can be specified instead of *question\_parse\_mode*
- is\_anonymous True, if the poll needs to be anonymous, defaults to True
- type Poll type, 'quiz' or 'regular', defaults to 'regular'
- allows\_multiple\_answers True, if the poll allows multiple answers, ignored for polls in quiz mode, defaults to False

- correct\_option\_id 0-based identifier of the correct answer option, required for polls in quiz mode
- **explanation** 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** Mode for parsing entities in the explanation. See formatting options for more details.
- **explanation\_entities** A JSON-serialized list of special entities that appear in the poll explanation. It can be specified instead of *explanation\_parse\_mode*
- **open\_period** Amount of time in seconds the poll will be active after creation, 5-600. Can't be used together with *close\_date*.
- **close\_date** 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** Pass True if the poll needs to be immediately closed. This can be useful for poll preview.
- disable\_notification Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- **message\_effect\_id** Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found

instance of method aiogram.methods.send\_poll.SendPoll

answer\_dice(business\_connection\_id: Optional[str] = None, message\_thread\_id: Optional[int] = None, emoji: Optional[str] = None, disable\_notification: Optional[bool] = None, protect\_content: Optional[Union[bool, Default]] = <Default('protect\_content')>, message\_effect\_id: Optional[str] = None, reply\_parameters: Optional[ReplyParameters] = None, reply\_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None, allow\_sending\_without\_reply: Optional[bool] = None, reply\_to\_message\_id: Optional[int] = None, \*\*kwargs: Any) → SendDice

Shortcut for method aiogram.methods.send\_dice.SendDice will automatically fill method attributes:

• chat\_id

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

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

#### **Parameters**

• business\_connection\_id — Unique identifier of the business connection on behalf of which the message will be sent

- message\_thread\_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **emoji** 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 Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send\_dice.SendDice

reply\_dice(business\_connection\_id: Optional[str] = None, message\_thread\_id: Optional[int] = None, emoji: Optional[str] = None, disable\_notification: Optional[bool] = None, protect\_content: Optional[Union[bool, Default]] = <Default('protect\_content') >, message\_effect\_id: Optional[str] = None, reply\_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None, allow\_sending\_without\_reply: Optional[bool] = None, \*\*kwargs: Any) → SendDice

Shortcut for method aiogram.methods.send\_dice.SendDice will automatically fill method attributes:

- chat\_id
- reply\_parameters

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

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

# **Parameters**

- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- **message\_thread\_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **emoji** 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 Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding

- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found

instance of method aiogram.methods.send\_dice.SendDice

Shortcut for method aiogram.methods.send\_sticker.SendSticker will automatically fill method attributes:

• chat\_id

Use this method to send static .WEBP, animated .TGS, or video .WEBM stickers. On success, the sent aiogram.types.message.Message is returned.

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

- **sticker** Sticker to send. Pass a file\_id as String to send a file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a .WEBP sticker from the Internet, or upload a new .WEBP, .TGS, or .WEBM sticker using multipart/formdata. *More information on Sending Files* ». Video and animated stickers can't be sent via an HTTP URL.
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- **message\_thread\_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- emoji Emoji associated with the sticker; only for just uploaded stickers
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found

• reply\_to\_message\_id - If the message is a reply, ID of the original message

#### Returns

instance of method aiogram.methods.send\_sticker.SendSticker

Shortcut for method <code>aiogram.methods.send\_sticker.SendSticker</code> will automatically fill method attributes:

- chat id
- reply\_parameters

Use this method to send static .WEBP, animated .TGS, or video .WEBM stickers. On success, the sent aiogram.types.message.Message is returned.

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

#### **Parameters**

- **sticker** Sticker to send. Pass a file\_id as String to send a file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a .WEBP sticker from the Internet, or upload a new .WEBP, .TGS, or .WEBM sticker using multipart/form-data. *More information on Sending Files* ». Video and animated stickers can't be sent via an HTTP URL.
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- **message\_thread\_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- emoji Emoji associated with the sticker; only for just uploaded stickers
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found

# Returns

instance of method aiogram.methods.send\_sticker.SendSticker

answer\_venue(latitude: float, longitude: float, title: str, address: str, business\_connection\_id: Optional[str] = None, message\_thread\_id: Optional[int] = None, foursquare\_id: Optional[str] = None, foursquare\_type: Optional[str] = None, google\_place\_id: Optional[str] = None, google\_place\_type: Optional[str] = None, disable\_notification: Optional[bool] = None, protect\_content: Optional[Union[bool, Default]] = <Default('protect\_content')>, message\_effect\_id: Optional[str] = None, reply\_parameters: Optional[ReplyParameters] = None, reply\_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None, allow\_sending\_without\_reply: Optional[bool] = None, reply\_to\_message\_id: Optional[int] = None, \*\*kwargs: Any) → SendVenue

Shortcut for method aiogram.methods.send\_venue.SendVenue will automatically fill method attributes:

## • chat\_id

Use this method to send information about a venue. On success, the sent aiogram.types.message. Message is returned.

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

#### **Parameters**

- latitude Latitude of the venue
- **longitude** Longitude of the venue
- title Name of the venue
- address Address of the venue
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- message\_thread\_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **foursquare\_id** Foursquare identifier of the venue
- **foursquare\_type** Foursquare type of the venue, if known. (For example, 'arts\_entertainment/default', 'arts\_entertainment/aquarium' or 'food/icecream'.)
- google\_place\_id Google Places identifier of the venue
- google\_place\_type Google Places type of the venue. (See supported types.)
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_parameters** Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

# Returns

 $instance\ of\ method\ aiogram.methods.send\_venue.SendVenue$ 

reply\_venue(latitude: float, longitude: float, title: str, address: str, business\_connection\_id: Optional[str] = None, message\_thread\_id: Optional[int] = None, foursquare\_id: Optional[str] = None, foursquare\_type: Optional[str] = None, google\_place\_id: Optional[str] = None, google\_place\_type: Optional[str] = None, disable\_notification: Optional[bool] = None, protect\_content: Optional[Union[bool, Default]] = <Default('protect\_content')>, message\_effect\_id: Optional[str] = None, reply\_markup:
Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None, allow\_sending\_without\_reply: Optional[bool] = None, \*\*kwargs: Any) → SendVenue

Shortcut for method aiogram.methods.send\_venue.SendVenue will automatically fill method attributes:

- chat\_id
- reply\_parameters

Use this method to send information about a venue. On success, the sent aiogram.types.message. Message is returned.

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

## **Parameters**

- latitude Latitude of the venue
- longitude Longitude of the venue
- title Name of the venue
- address Address of the venue
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- message\_thread\_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **foursquare\_id** Foursquare identifier of the venue
- **foursquare\_type** Foursquare type of the venue, if known. (For example, 'arts\_entertainment/default', 'arts\_entertainment/aquarium' or 'food/icecream'.)
- google\_place\_id Google Places identifier of the venue
- **google\_place\_type** Google Places type of the venue. (See supported types.)
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found

# Returns

instance of method aiogram.methods.send\_venue.SendVenue

Shortcut for method aiogram.methods.send\_video.SendVideo will automatically fill method attributes:

• chat\_id

Use this method to send video files, Telegram clients support MPEG4 videos (other formats may be sent as aiogram.types.document.Document). On success, the sent aiogram.types.message.Message is returned. Bots can currently send video files of up to 50 MB in size, this limit may be changed in the future.

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

- **video** Video to send. Pass a file\_id as String to send a video that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a video from the Internet, or upload a new video using multipart/form-data. *More information on Sending Files* »
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- **message\_thread\_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- duration Duration of sent video in seconds
- width Video width
- **height** Video height
- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file\_attach\_name>' if the thumbnail was uploaded using multipart/form-data under <file\_attach\_name>. *More information on Sending Files*
- **caption** Video caption (may also be used when resending videos by *file\_id*), 0-1024 characters after entities parsing
- parse\_mode Mode for parsing entities in the video caption. See formatting options for more details.
- **caption\_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse\_mode*

- **show\_caption\_above\_media** Pass True, if the caption must be shown above the message media
- has\_spoiler Pass True if the video needs to be covered with a spoiler animation
- **supports\_streaming** Pass True if the uploaded video is suitable for streaming
- disable\_notification Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

#### Returns

instance of method aiogram.methods.send\_video.SendVideo

Shortcut for method aiogram.methods.send\_video.SendVideo will automatically fill method attributes:

- chat\_id
- reply\_parameters

Use this method to send video files, Telegram clients support MPEG4 videos (other formats may be sent as aiogram.types.document.Document). On success, the sent aiogram.types.message.Message is returned. Bots can currently send video files of up to 50 MB in size, this limit may be changed in the future.

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

#### **Parameters**

video – Video to send. Pass a file\_id as String to send a video that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a video from the Internet, or upload a new video using multipart/form-data. More information on Sending Files »

- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- **message\_thread\_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- duration Duration of sent video in seconds
- width Video width
- height Video height
- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file\_attach\_name>' if the thumbnail was uploaded using multipart/form-data under <file\_attach\_name>. *More information on Sending Files*
- **caption** Video caption (may also be used when resending videos by *file\_id*), 0-1024 characters after entities parsing
- parse\_mode Mode for parsing entities in the video caption. See formatting options for more details.
- **caption\_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse\_mode*
- **show\_caption\_above\_media** Pass True, if the caption must be shown above the message media
- has\_spoiler Pass True if the video needs to be covered with a spoiler animation
- **supports\_streaming** Pass True if the uploaded video is suitable for streaming
- disable\_notification Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found

#### Returns

instance of method aiogram.methods.send\_video.SendVideo

Shortcut for method aiogram.methods.send\_video\_note.SendVideoNote will automatically fill method attributes:

• chat\_id

As of v.4.0, Telegram clients support rounded square MPEG4 videos of up to 1 minute long. Use this method to send video messages. On success, the sent aiogram.types.message.Message is returned.

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

#### **Parameters**

- **video\_note** Video note to send. Pass a file\_id as String to send a video note that exists on the Telegram servers (recommended) or upload a new video using multipart/form-data. *More information on Sending Files* ». Sending video notes by a URL is currently unsupported
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- message\_thread\_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- duration Duration of sent video in seconds
- length Video width and height, i.e. diameter of the video message
- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file\_attach\_name>' if the thumbnail was uploaded using multipart/form-data under <file\_attach\_name>. *More information on Sending Files*
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

#### Returns

instance of method aiogram.methods.send\_video\_note.SendVideoNote

```
reply_video_note (video_note: Union[InputFile, str], business_connection_id: Optional[str] = None, message_thread_id: Optional[int] = None, duration: Optional[int] = None, length: Optional[int] = None, thumbnail: Optional[InputFile] = None, disable_notification: Optional[bool] = None, protect_content: Optional[Union[bool, Default]] = <Default('protect_content')>, message_effect_id: Optional[str] = None, reply_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None, allow_sending_without_reply: Optional[bool] = None, **kwargs: Any) → SendVideoNote
```

Shortcut for method  $aiogram.methods.send\_video\_note.SendVideoNote$  will automatically fill method attributes:

- chat\_id
- reply\_parameters

As of v.4.0, Telegram clients support rounded square MPEG4 videos of up to 1 minute long. Use this method to send video messages. On success, the sent aiogram.types.message.Message is returned.

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

#### **Parameters**

- **video\_note** Video note to send. Pass a file\_id as String to send a video note that exists on the Telegram servers (recommended) or upload a new video using multipart/form-data. *More information on Sending Files* ». Sending video notes by a URL is currently unsupported
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- message\_thread\_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- duration Duration of sent video in seconds
- length Video width and height, i.e. diameter of the video message
- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file\_attach\_name>' if the thumbnail was uploaded using multipart/form-data under <file\_attach\_name>. *More information on Sending Files*»
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found

# Returns

instance of method aiogram.methods.send\_video\_note.SendVideoNote

Shortcut for method aiogram.methods.send\_voice.SendVoice will automatically fill method attributes:

#### • chat\_id

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, or in .MP3 format, or in .M4A format (other formats may be sent as <code>aiogram.types.audio.Audio</code> or <code>aiogram.types.document.Document</code>). On success, the sent <code>aiogram.types.message.Message</code> is returned. Bots can currently send voice messages of up to 50 MB in size, this limit may be changed in the future.

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

#### **Parameters**

- **voice** Audio file to send. Pass a file\_id as String to send a file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a file from the Internet, or upload a new one using multipart/form-data. *More information on Sending Files* »
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- message\_thread\_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- caption Voice message caption, 0-1024 characters after entities parsing
- parse\_mode Mode for parsing entities in the voice message caption. See formatting options for more details.
- **caption\_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse\_mode*
- duration Duration of the voice message in seconds
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found

• reply\_to\_message\_id - If the message is a reply, ID of the original message

#### Returns

instance of method aiogram.methods.send\_voice.SendVoice

Shortcut for method aiogram.methods.send\_voice.SendVoice will automatically fill method attributes:

- chat\_id
- reply\_parameters

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, or in .MP3 format, or in .M4A format (other formats may be sent as <code>aiogram.types.audio.Audio</code> or <code>aiogram.types.document.Document</code>). On success, the sent <code>aiogram.types.message</code> is returned. Bots can currently send voice messages of up to 50 MB in size, this limit may be changed in the future.

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

#### **Parameters**

- **voice** Audio file to send. Pass a file\_id as String to send a file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a file from the Internet, or upload a new one using multipart/form-data. *More information on Sending Files* »
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- message\_thread\_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- caption Voice message caption, 0-1024 characters after entities parsing
- parse\_mode Mode for parsing entities in the voice message caption. See formatting options for more details.
- caption\_entities A JSON-serialized list of special entities that appear in the caption, which can be specified instead of parse mode
- duration Duration of the voice message in seconds
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

• allow\_sending\_without\_reply - Pass True if the message should be sent even if the specified replied-to message is not found

#### Returns

instance of method aiogram.methods.send\_voice.SendVoice

Shortcut for method aiogram.methods.send\_paid\_media.SendPaidMedia will automatically fill method attributes:

• chat id

Use this method to send paid media. On success, the sent aiogram.types.message.Message is returned.

Source: https://core.telegram.org/bots/api#sendpaidmedia

#### **Parameters**

- star\_count The number of Telegram Stars that must be paid to buy access to the media;
   1-2500
- media A JSON-serialized array describing the media to be sent; up to 10 items
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- **payload** Bot-defined paid media payload, 0-128 bytes. This will not be displayed to the user, use it for your internal processes.
- **caption** Media caption, 0-1024 characters after entities parsing
- parse\_mode Mode for parsing entities in the media caption. See formatting options for more details.
- **caption\_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse\_mode*
- **show\_caption\_above\_media** Pass True, if the caption must be shown above the message media
- disable\_notification Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

# Returns

instance of method aiogram.methods.send\_paid\_media.SendPaidMedia

model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

Shortcut for method aiogram.methods.send\_paid\_media.SendPaidMedia will automatically fill method attributes:

- · chat id
- reply\_parameters

Use this method to send paid media. On success, the sent aiogram.types.message.Message is returned.

Source: https://core.telegram.org/bots/api#sendpaidmedia

#### **Parameters**

- star\_count The number of Telegram Stars that must be paid to buy access to the media;
   1-2500
- media A JSON-serialized array describing the media to be sent; up to 10 items
- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be sent
- payload Bot-defined paid media payload, 0-128 bytes. This will not be displayed to the user, use it for your internal processes.
- caption Media caption, 0-1024 characters after entities parsing
- parse\_mode Mode for parsing entities in the media caption. See formatting options for more details.
- **caption\_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse\_mode*
- **show\_caption\_above\_media** Pass True, if the caption must be shown above the message media
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

### Returns

instance of method aiogram.methods.send\_paid\_media.SendPaidMedia

```
 \textbf{as\_reply\_parameters}(allow\_sending\_without\_reply: Optional[Union[bool, Default]] = \\ < Default('allow\_sending\_without\_reply') >, quote: Optional[str] = None, \\ quote\_parse\_mode: Optional[Union[str, Default]] = < Default('parse\_mode') >, \\ quote\_entities: Optional[List[MessageEntity]] = None, quote\_position: \\ Optional[int] = None) \rightarrow ReplyParameters
```

### InlineKeyboardButton

class aiogram.types.inline\_keyboard\_button.InlineKeyboardButton(\*, text: str, url: str | None =

None, callback\_data: str | None
= None, web\_app: WebAppInfo
| None = None, login\_url:
LoginUrl | None = None,
switch\_inline\_query: str | None
= None,
switch\_inline\_query\_current\_chat:
str | None = None,
switch\_inline\_query\_chosen\_chat:
SwitchInlineQueryChosenChat |
None = None, callback\_game:
CallbackGame | None = None,
pay: bool | None = None,
\*\*extra\_data: Any)

This object represents one button of an inline keyboard. Exactly one of the optional fields must be used to specify type of the button.

Source: https://core.telegram.org/bots/api#inlinekeyboardbutton

#### text: str

Label text on the button

# url: str | None

*Optional*. HTTP or tg:// URL to be opened when the button is pressed. Links tg://user?id=<user\_id> can be used to mention a user by their identifier without using a username, if this is allowed by their privacy settings.

#### callback\_data: str | None

Optional. Data to be sent in a callback query to the bot when the button is pressed, 1-64 bytes

# web\_app: WebAppInfo | None

Optional. Description of the Web App that will be launched when the user presses the button. The Web App will be able to send an arbitrary message on behalf of the user using the method <code>aiogram.methods.answer\_web\_app\_query.AnswerWebAppQuery</code>. Available only in private chats between a user and the bot. Not supported for messages sent on behalf of a Telegram Business account.

#### login\_url: LoginUrl | None

*Optional*. An HTTPS URL used to automatically authorize the user. Can be used as a replacement for the Telegram Login Widget.

# model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

# switch\_inline\_query: str | None

*Optional.* If set, pressing the button will prompt the user to select one of their chats, open that chat and insert the bot's username and the specified inline query in the input field. May be empty, in which case just the bot's username will be inserted. Not supported for messages sent on behalf of a Telegram Business account.

```
switch_inline_query_current_chat: str | None
```

*Optional*. If set, pressing the button will insert the bot's username and the specified inline query in the current chat's input field. May be empty, in which case only the bot's username will be inserted.

```
switch_inline_query_chosen_chat: SwitchInlineQueryChosenChat | None
```

*Optional*. If set, pressing the button will prompt the user to select one of their chats of the specified type, open that chat and insert the bot's username and the specified inline query in the input field. Not supported for messages sent on behalf of a Telegram Business account.

```
callback_game: CallbackGame | None
```

Optional. Description of the game that will be launched when the user presses the button.

```
pay: bool | None
```

*Optional*. Specify True, to send a Pay button. Substrings 'and 'XTR' in the buttons's text will be replaced with a Telegram Star icon.

# **InlineKeyboardMarkup**

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

Source: https://core.telegram.org/bots/api#inlinekeyboardmarkup

```
inline_keyboard: List[List[InlineKeyboardButton]]
```

Array of button rows, each represented by an Array of aiogram.types.inline\_keyboard\_button.
InlineKeyboardButton objects

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

# InputFile

```
class aiogram.types.input_file.InputFile(filename: str | None = None, chunk_size: int = 65536)
```

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.

Source: https://core.telegram.org/bots/api#inputfile

```
abstract async read(bot: Bot) \rightarrow AsyncGenerator[bytes, None]
```

```
class aiogram.types.input_file.BufferedInputFile(file: bytes, filename: str, chunk_size: int = 65536)
```

```
classmethod from_file(path: str \mid Path, filename: str \mid None = None, chunk_size: int = 65536) <math>\rightarrow BufferedInputFile
```

Create buffer from file

#### **Parameters**

- path Path to file
- filename Filename to be propagated to telegram. By default, will be parsed from path

```
• chunk_size – Uploading chunk size
```

#### Returns

instance of BufferedInputFile

```
async read(bot: Bot) \rightarrow AsyncGenerator[bytes, None]
```

```
async read(bot: Bot) \rightarrow AsyncGenerator[bytes, None]
```

class aiogram.types.input\_file.URLInputFile( $url: str, headers: Dict[str, Any] \mid None = None, filename: str \mid None = None, chunk_size: int = 65536, timeout: int = 30, bot: 'Bot' \mid None = None)$ 

**async read**(bot: Bot)  $\rightarrow$  AsyncGenerator[bytes, None]

# **InputMedia**

class aiogram.types.input\_media.InputMedia(\*\*extra\_data: Any)

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

- aiogram.types.input\_media\_animation.InputMediaAnimation
- aiogram.types.input\_media\_document.InputMediaDocument
- aiogram.types.input\_media\_audio.InputMediaAudio
- aiogram.types.input\_media\_photo.InputMediaPhoto
- aiogram.types.input\_media\_video.InputMediaVideo

Source: https://core.telegram.org/bots/api#inputmedia

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

# **InputMediaAnimation**

```
class aiogram.types.input_media_animation.InputMediaAnimation(*, type: ~typ-
                                                                             ing.Literal[InputMediaType.ANIMATION]
                                                                             = InputMediaType.ANIMATION,
                                                                             media: str |
                                                                             ~aiogram.types.input_file.InputFile,
                                                                             thumbnail:
                                                                             ~aiogram.types.input file.InputFile
                                                                             | None = None, caption: str | None
                                                                             = None, parse mode: str
                                                                             ~aiogram.client.default.Default |
                                                                             None = < Default('parse\_mode')>,
                                                                             caption_entities: ~typ-
                                                                             ing.List[~aiogram.types.message_entity.MessageEntit
                                                                             | None = None,
                                                                             show_caption_above_media: bool |
                                                                             ~aiogram.client.default.Default |
                                                                             None = < De-
                                                                             fault('show caption above media')>,
                                                                             width: int \mid None = None, height:
                                                                             int \mid None = None, duration: int \mid
                                                                             None = None, has\_spoiler: bool |
                                                                             None = None, **extra data:
                                                                             ~typing.Any)
```

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

Source: https://core.telegram.org/bots/api#inputmediaanimation

### type: Literal[InputMediaType.ANIMATION]

Type of the result, must be animation

### media: str | InputFile

File to send. Pass a file\_id to send a file that exists on the Telegram servers (recommended), pass an HTTP URL for Telegram to get a file from the Internet, or pass 'attach://<file\_attach\_name>' to upload a new one using multipart/form-data under <file\_attach\_name> name. *More information on Sending Files* »

### thumbnail: InputFile | None

Optional. Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported serverside. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file\_attach\_name>' if the thumbnail was uploaded using multipart/form-data under <file\_attach\_name>. More information on Sending Files >>

#### caption: str | None

Optional. Caption of the animation to be sent, 0-1024 characters after entities parsing

# parse\_mode: str | Default | None

Optional. Mode for parsing entities in the animation caption. See formatting options for more details.

### caption\_entities: List[MessageEntity] | None

Optional. List of special entities that appear in the caption, which can be specified instead of parse\_mode

### model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

### show\_caption\_above\_media: bool | Default | None

Optional. Pass True, if the caption must be shown above the message media

width: int | None

Optional. Animation width

height: int | None

Optional. Animation height

duration: int | None

Optional. Animation duration in seconds

has\_spoiler: bool | None

Optional. Pass True if the animation needs to be covered with a spoiler animation

### InputMediaAudio

```
class aiogram.types.input_media_audio.InputMediaAudio(*, type:
```

```
~typing.Literal[InputMediaType.AUDIO] =
InputMediaType.AUDIO, media: str |
~aiogram.types.input_file.InputFile,
thumbnail:
~aiogram.types.input_file.InputFile | None =
None, caption: str | None = None,
parse_mode: str |
~aiogram.client.default.Default | None =
<Default('parse_mode')>, caption_entities:
~typ-
ing.List[~aiogram.types.message_entity.MessageEntity]
| None = None, duration: int | None = None,
performer: str | None = None, title: str | None
= None, **extra_data: ~typing.Any)
```

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

Source: https://core.telegram.org/bots/api#inputmediaaudio

type: Literal[InputMediaType.AUDI0]

Type of the result, must be audio

media: str | InputFile

File to send. Pass a file\_id to send a file that exists on the Telegram servers (recommended), pass an HTTP URL for Telegram to get a file from the Internet, or pass 'attach://<file\_attach\_name>' to upload a new one using multipart/form-data under <file\_attach\_name> name. *More information on Sending Files* »

# thumbnail: InputFile | None

Optional. Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported serverside. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file\_attach\_name>' if the thumbnail was uploaded using multipart/form-data under <file\_attach\_name>. More information on Sending Files >>

```
caption: str | None
     Optional. Caption of the audio to be sent, 0-1024 characters after entities parsing
parse_mode: str | Default | None
     Optional. Mode for parsing entities in the audio caption. See formatting options for more details.
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
     A dictionary of computed field names and their corresponding ComputedFieldInfo objects.
model_post_init(context: Any, /) \rightarrow None
     We need to both initialize private attributes and call the user-defined model_post_init method.
caption_entities: List[MessageEntity] | None
     Optional. List of special entities that appear in the caption, which can be specified instead of parse_mode
duration: int | None
     Optional. Duration of the audio in seconds
performer: str | None
     Optional. Performer of the audio
title: str | None
     Optional. Title of the audio
```

# InputMediaDocument

```
class aiogram.types.input_media_document.InputMediaDocument(*, type: ~typ-
                                                                         ing.Literal[InputMediaType.DOCUMENT]
                                                                         = InputMediaType.DOCUMENT,
                                                                         media: str |
                                                                         ~aiogram.types.input_file.InputFile,
                                                                         thumbnail:
                                                                         ~aiogram.types.input_file.InputFile |
                                                                         None = None, caption: str \mid None =
                                                                         None, parse_mode: str |
                                                                         ~aiogram.client.default.Default | None
                                                                         = < Default('parse\_mode')>,
                                                                         caption entities: ~typ-
                                                                         ing.List[~aiogram.types.message_entity.MessageEntity]
                                                                         | None = None,
                                                                         disable content type detection: bool
                                                                         | None = None, **extra_data:
                                                                         ~typing.Any)
```

Represents a general file to be sent.

Source: https://core.telegram.org/bots/api#inputmediadocument

# type: Literal[InputMediaType.DOCUMENT]

Type of the result, must be document

```
media: str | InputFile
```

File to send. Pass a file\_id to send a file that exists on the Telegram servers (recommended), pass an HTTP URL for Telegram to get a file from the Internet, or pass 'attach://<file\_attach\_name>' to upload a new one using multipart/form-data under <file\_attach\_name> name. More information on Sending Files >>

# thumbnail: InputFile | None

*Optional*. Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported serverside. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file\_attach\_name>' if the thumbnail was uploaded using multipart/form-data under <file\_attach\_name>. *More information on Sending Files* "

### caption: str | None

Optional. Caption of the document to be sent, 0-1024 characters after entities parsing

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
parse_mode: str | Default | None
```

Optional. Mode for parsing entities in the document caption. See formatting options for more details.

```
caption_entities: List[MessageEntity] | None
```

Optional. List of special entities that appear in the caption, which can be specified instead of parse\_mode

```
disable_content_type_detection: bool | None
```

*Optional*. Disables automatic server-side content type detection for files uploaded using multipart/form-data. Always True, if the document is sent as part of an album.

# InputMediaPhoto

class aiogram.types.input\_media\_photo.InputMediaPhoto(\*, type:

```
~typing.Literal[InputMediaType.PHOTO] =
InputMediaType.PHOTO, media: str |
~aiogram.types.input_file.InputFile, caption:
str | None = None, parse_mode: str |
~aiogram.client.default.Default | None =
<Default('parse_mode')>, caption_entities:
~typ-
ing.List[~aiogram.types.message_entity.MessageEntity]
| None = None, show_caption_above_media:
bool | ~aiogram.client.default.Default | None
= <Default('show_caption_above_media')>,
has_spoiler: bool | None = None,
**extra_data: ~typing.Any)
```

Represents a photo to be sent.

Source: https://core.telegram.org/bots/api#inputmediaphoto

# type: Literal[InputMediaType.PHOTO]

Type of the result, must be photo

```
media: str | InputFile
```

File to send. Pass a file\_id to send a file that exists on the Telegram servers (recommended), pass an HTTP URL for Telegram to get a file from the Internet, or pass 'attach://<file\_attach\_name>' to upload a new one using multipart/form-data under <file\_attach\_name> name. *More information on Sending Files* »

```
caption: str | None

Optional. Caption of the photo to be sent, 0-1024 characters after entities parsing

parse_mode: str | Default | None

Optional. Mode for parsing entities in the photo caption. See formatting options for more details.

model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

model_post_init(context: Any, /) → None

We need to both initialize private attributes and call the user-defined model_post_init method.

caption_entities: List[MessageEntity] | None

Optional. List of special entities that appear in the caption, which can be specified instead of parse_mode

show_caption_above_media: bool | Default | None

Optional. Pass True, if the caption must be shown above the message media

has_spoiler: bool | None

Optional. Pass True if the photo needs to be covered with a spoiler animation
```

# InputMediaVideo

```
class aiogram.types.input_media_video.InputMediaVideo(*, type:
                                                                    ~typing.Literal[InputMediaType.VIDEO] =
                                                                    InputMediaType.VIDEO, media: str |
                                                                    ~aiogram.types.input_file.InputFile,
                                                                    thumbnail:
                                                                    ~aiogram.types.input_file.InputFile | None =
                                                                    None, caption: str \mid None = None,
                                                                    parse_mode: str |
                                                                    ~aiogram.client.default.Default | None =
                                                                    <Default('parse_mode')>, caption_entities:
                                                                    ~tvp-
                                                                    ing.List[~aiogram.types.message_entity.MessageEntity]
                                                                    | None = None, show_caption_above_media:
                                                                    bool | ~aiogram.client.default.Default | None
                                                                    = < Default('show caption above media')>,
                                                                    width: int | None = None, height: int | None =
                                                                    None, duration: int \mid None = None,
                                                                    supports streaming: bool \mid None = None,
                                                                    has\_spoiler: bool \mid None = None,
```

Represents a video to be sent.

Source: https://core.telegram.org/bots/api#inputmediavideo

type: Literal[InputMediaType.VIDEO]

Type of the result, must be video

media: str | InputFile

File to send. Pass a file\_id to send a file that exists on the Telegram servers (recommended), pass an HTTP URL for Telegram to get a file from the Internet, or pass 'attach://<file\_attach\_name>' to upload a new one using multipart/form-data under <file\_attach\_name> name. More information on Sending Files >>

\*\*extra\_data: ~typing.Any)

```
thumbnail: InputFile | None
```

*Optional*. Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported serverside. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file\_attach\_name>' if the thumbnail was uploaded using multipart/form-data under <file\_attach\_name>. *More information on Sending Files* "

caption: str | None

Optional. Caption of the video to be sent, 0-1024 characters after entities parsing

parse\_mode: str | Default | None

Optional. Mode for parsing entities in the video caption. See formatting options for more details.

caption\_entities: List[MessageEntity] | None

Optional. List of special entities that appear in the caption, which can be specified instead of parse\_mode

model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

 $model_post_init(context: Any, /) \rightarrow None$ 

We need to both initialize private attributes and call the user-defined model\_post\_init method.

show\_caption\_above\_media: bool | Default | None

Optional. Pass True, if the caption must be shown above the message media

width: int | None

Optional. Video width

height: int | None

Optional. Video height

duration: int | None

Optional. Video duration in seconds

supports\_streaming: bool | None

Optional. Pass True if the uploaded video is suitable for streaming

has\_spoiler: bool | None

Optional. Pass True if the video needs to be covered with a spoiler animation

# **InputPaidMedia**

class aiogram.types.input\_paid\_media.InputPaidMedia(\*\*extra\_data: Any)

This object describes the paid media to be sent. Currently, it can be one of

- aiogram.types.input\_paid\_media\_photo.InputPaidMediaPhoto
- aiogram.types.input\_paid\_media\_video.InputPaidMediaVideo

Source: https://core.telegram.org/bots/api#inputpaidmedia

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

 $model_post_init(context: Any, /) \rightarrow None$ 

We need to both initialize private attributes and call the user-defined model\_post\_init method.

# InputPaidMediaPhoto

eral[InputPaidMediaType.PHOTO]
= InputPaidMediaType.PHOTO,
media: str | InputFile,
\*\*extra\_data: Any)

The paid media to send is a photo.

Source: https://core.telegram.org/bots/api#inputpaidmediaphoto

type: Literal[InputPaidMediaType.PHOTO]

Type of the media, must be photo

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model post init method.

```
media: str | InputFile
```

File to send. Pass a file\_id to send a file that exists on the Telegram servers (recommended), pass an HTTP URL for Telegram to get a file from the Internet, or pass 'attach://<file\_attach\_name>' to upload a new one using multipart/form-data under <file\_attach\_name> name. *More information on Sending Files* »

# InputPaidMediaVideo

class aiogram.types.input\_paid\_media\_video.InputPaidMediaVideo(\*, type: Lit-

eral[InputPaidMediaType.VIDEO]
= InputPaidMediaType.VIDEO,
media: str | InputFile, thumbnail:
InputFile | str | None = None,
width: int | None = None, height:
int | None = None, duration: int |
None = None,
supports\_streaming: bool | None
= None, \*\*extra\_data: Any)

The paid media to send is a video.

Source: https://core.telegram.org/bots/api#inputpaidmediavideo

type: Literal[InputPaidMediaType.VIDE0]

Type of the media, must be video

```
media: str | InputFile
```

File to send. Pass a file\_id to send a file that exists on the Telegram servers (recommended), pass an HTTP URL for Telegram to get a file from the Internet, or pass 'attach://<file\_attach\_name>' to upload a new one using multipart/form-data under <file\_attach\_name> name. More information on Sending Files >>

```
thumbnail: InputFile | str | None
```

Optional. Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file\_attach\_name>' if

the thumbnail was uploaded using multipart/form-data under <file\_attach\_name>. *More information on Sending Files* »

### width: int | None

Optional. Video width

### model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

## $model_post_init(context: Any, /) \rightarrow None$

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### height: int | None

Optional. Video height

### duration: int | None

Optional. Video duration in seconds

# supports\_streaming: bool | None

Optional. Pass True if the uploaded video is suitable for streaming

# InputPollOption

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

Source: https://core.telegram.org/bots/api#inputpolloption

# text: str

Option text, 1-100 characters

#### text\_parse\_mode: str | Default | None

*Optional*. Mode for parsing entities in the text. See formatting options for more details. Currently, only custom emoji entities are allowed

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
text_entities: List[MessageEntity] | None
```

*Optional*. A JSON-serialized list of special entities that appear in the poll option text. It can be specified instead of *text\_parse\_mode* 

# KeyboardButton

class aiogram.types.keyboard\_button.KeyboardButton(\*, text: str, request\_users:

KeyboardButtonRequestUsers | None = None, request\_chat: KeyboardButtonRequestChat | None = None, request\_contact: bool | None = None, request\_location: bool | None = None, request\_poll: KeyboardButtonPollType | None = None, web\_app: WebAppInfo | None = None, request\_user: KeyboardButtonRequestUser | None = None, \*\*extra\_data: Any)

This object represents one button of the reply keyboard. At most one of the optional fields must be used to specify type of the button. For simple text buttons, *String* can be used instead of this object to specify the button text. **Note:** *request\_users* and *request\_chat* options will only work in Telegram versions released after 3 February, 2023. Older clients will display *unsupported message*.

Source: https://core.telegram.org/bots/api#keyboardbutton

#### text: str

Text of the button. If none of the optional fields are used, it will be sent as a message when the button is pressed

# request\_users: KeyboardButtonRequestUsers | None

*Optional.* If specified, pressing the button will open a list of suitable users. Identifiers of selected users will be sent to the bot in a 'users\_shared' service message. Available in private chats only.

#### request\_chat: KeyboardButtonRequestChat | None

*Optional.* If specified, pressing the button will open a list of suitable chats. Tapping on a chat will send its identifier to the bot in a 'chat\_shared' service message. Available in private chats only.

### request\_contact: bool | None

*Optional*. If True, the user's phone number will be sent as a contact when the button is pressed. Available in private chats only.

# model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

### $model_post_init(context: Any, /) \rightarrow None$

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### request\_location: bool | None

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

# request\_poll: KeyboardButtonPollType | None

*Optional*. If specified, the user will be asked to create a poll and send it to the bot when the button is pressed. Available in private chats only.

#### web\_app: WebAppInfo | None

*Optional*. If specified, the described Web App will be launched when the button is pressed. The Web App will be able to send a 'web app data' service message. Available in private chats only.

### request\_user: KeyboardButtonRequestUser | None

*Optional.* If specified, pressing the button will open a list of suitable users. Tapping on any user will send their identifier to the bot in a 'user shared' service message. Available in private chats only.

Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023

\*\*extra data:

Any)

# KeyboardButtonPollType

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

Source: https://core.telegram.org/bots/api#keyboardbuttonpolltype

```
type: str | None
```

*Optional.* If *quiz* is passed, the user will be allowed to create only polls in the quiz mode. If *regular* is passed, only regular polls will be allowed. Otherwise, the user will be allowed to create a poll of any type.

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

### KeyboardButtonRequestChat

```
class aiogram.types.keyboard_button_request_chat.KeyboardButtonRequestChat(*, request_id: int,
                                                                                          chat_is_channel:
                                                                                          bool,
                                                                                          chat_is_forum:
                                                                                          bool | None =
                                                                                          None,
                                                                                          chat_has_username:
                                                                                          bool | None =
                                                                                          None,
                                                                                          chat is created:
                                                                                          bool | None =
                                                                                          None,
                                                                                          user_administrator_rights:
                                                                                          ChatAdministra-
                                                                                          torRights | None =
                                                                                          None,
                                                                                          bot_administrator_rights:
                                                                                          ChatAdministra-
                                                                                          torRights | None =
                                                                                          None,
                                                                                          bot_is_member:
                                                                                          bool | None =
                                                                                          None,
                                                                                          request title: bool
                                                                                          | None = None, re-
                                                                                          quest username:
                                                                                          bool | None =
                                                                                          None.
                                                                                          request_photo:
                                                                                          bool | None =
                                                                                          None,
```

This object defines the criteria used to request a suitable chat. Information about the selected chat will be shared with the bot when the corresponding button is pressed. The bot will be granted requested rights in the chat if appropriate. More about requesting chats ».

Source: https://core.telegram.org/bots/api#keyboardbuttonrequestchat

### request\_id: int

Signed 32-bit identifier of the request, which will be received back in the aiogram.types.chat\_shared. ChatShared object. Must be unique within the message

#### chat\_is\_channel: bool

Pass True to request a channel chat, pass False to request a group or a supergroup chat.

#### chat\_is\_forum: bool | None

*Optional*. Pass True to request a forum supergroup, pass False to request a non-forum chat. If not specified, no additional restrictions are applied.

### chat\_has\_username: bool | None

*Optional*. Pass True to request a supergroup or a channel with a username, pass False to request a chat without a username. If not specified, no additional restrictions are applied.

### chat\_is\_created: bool | None

Optional. Pass True to request a chat owned by the user. Otherwise, no additional restrictions are applied.

# user\_administrator\_rights: ChatAdministratorRights | None

*Optional*. A JSON-serialized object listing the required administrator rights of the user in the chat. The rights must be a superset of *bot\_administrator\_rights*. If not specified, no additional restrictions are applied.

# model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

# $model_post_init(context: Any, /) \rightarrow None$

We need to both initialize private attributes and call the user-defined model\_post\_init method.

### bot\_administrator\_rights: ChatAdministratorRights | None

*Optional*. A JSON-serialized object listing the required administrator rights of the bot in the chat. The rights must be a subset of *user\_administrator\_rights*. If not specified, no additional restrictions are applied.

# bot\_is\_member: bool | None

*Optional*. Pass True to request a chat with the bot as a member. Otherwise, no additional restrictions are applied.

#### request\_title: bool | None

Optional. Pass True to request the chat's title

# request\_username: bool | None

Optional. Pass True to request the chat's username

### request\_photo: bool | None

Optional. Pass True to request the chat's photo

# KeyboardButtonRequestUser

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

Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023

Source: https://core.telegram.org/bots/api#keyboardbuttonrequestuser

### request\_id: int

Signed 32-bit identifier of the request, which will be received back in the aiogram.types.user\_shared. UserShared object. Must be unique within the message

### user\_is\_bot: bool | None

*Optional.* Pass True to request a bot, pass False to request a regular user. If not specified, no additional restrictions are applied.

# model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### user\_is\_premium: bool | None

*Optional*. Pass True to request a premium user, pass False to request a non-premium user. If not specified, no additional restrictions are applied.

### KeyboardButtonRequestUsers

 ${\bf class} \ {\bf aiogram.types.keyboard\_button\_request\_users.} {\bf KeyboardButtonRequestUsers} (*, {\it request\_id:}$ 

int, user is bot:  $bool \mid None =$ None. user\_is\_premium: bool | None = None. *max\_quantity: int* | *None* = None, request\_name: bool | None = None, request\_username: bool | None = None, request\_photo: bool | None = None. \*\*extra data:

Any)

This object defines the criteria used to request suitable users. Information about the selected users will be shared with the bot when the corresponding button is pressed. More about requesting users »

Source: https://core.telegram.org/bots/api#keyboardbuttonrequestusers

#### request\_id: int

Signed 32-bit identifier of the request that will be received back in the *aiogram.types.users\_shared*. *UsersShared* object. Must be unique within the message

### user\_is\_bot: bool | None

*Optional.* Pass True to request bots, pass False to request regular users. If not specified, no additional restrictions are applied.

### user\_is\_premium: bool | None

*Optional*. Pass True to request premium users, pass False to request non-premium users. If not specified, no additional restrictions are applied.

# max\_quantity: int | None

*Optional*. The maximum number of users to be selected; 1-10. Defaults to 1.

# model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding  ${\it ComputedFieldInfo}$  objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

# request\_name: bool | None

Optional. Pass True to request the users' first and last names

# request\_username: bool | None

Optional. Pass True to request the users' usernames

# request\_photo: bool | None

Optional. Pass True to request the users' photos

\*\*extra\_data: ~typing.Any)

# LinkPreviewOptions

```
class aiogram.types.link_preview_options.LinkPreviewOptions(*, is_disabled: bool |
                                                                           ~aiogram.client.default.Default | None
                                                                           = < De-
                                                                           fault('link_preview_is_disabled')>,
                                                                           url: str \mid None = None,
                                                                           prefer small media: bool |
                                                                           ~aiogram.client.default.Default | None
                                                                           = < De-
                                                                           fault('link_preview_prefer_small_media')>,
                                                                           prefer_large_media: bool |
                                                                           ~aiogram.client.default.Default | None
                                                                           = <De-
                                                                          fault('link preview prefer large media')>,
                                                                           show_above_text: bool |
                                                                           ~aiogram.client.default.Default | None
                                                                           = <De-
                                                                           fault('link_preview_show_above_text')>,
```

Describes the options used for link preview generation.

Source: https://core.telegram.org/bots/api#linkpreviewoptions

# is\_disabled: bool | Default | None

Optional. True, if the link preview is disabled

### url: str | None

Optional. URL to use for the link preview. If empty, then the first URL found in the message text will be used

# prefer\_small\_media: bool | Default | None

*Optional*. True, if the media in the link preview is supposed to be shrunk; ignored if the URL isn't explicitly specified or media size change isn't supported for the preview

# model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

# prefer\_large\_media: bool | Default | None

*Optional*. True, if the media in the link preview is supposed to be enlarged; ignored if the URL isn't explicitly specified or media size change isn't supported for the preview

### show\_above\_text: bool | Default | None

*Optional*. True, if the link preview must be shown above the message text; otherwise, the link preview will be shown below the message text

#### Location

```
class aiogram.types.location.Location(*, latitude: float, longitude: float, horizontal_accuracy: float | None = None, live_period: int | None = None, heading: int | None = None, proximity_alert_radius: int | None = None, **extra_data: Any)
```

This object represents a point on the map.

Source: https://core.telegram.org/bots/api#location

latitude: float

Latitude as defined by the sender

longitude: float

Longitude as defined by the sender

horizontal\_accuracy: float | None

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

model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

 $model_post_init(context: Any, /) \rightarrow None$ 

We need to both initialize private attributes and call the user-defined model\_post\_init method.

live\_period: int | None

*Optional*. Time relative to the message sending date, during which the location can be updated; in seconds. For active live locations only.

heading: int | None

Optional. The direction in which user is moving, in degrees; 1-360. For active live locations only.

proximity\_alert\_radius: int | None

*Optional*. The maximum distance for proximity alerts about approaching another chat member, in meters. For sent live locations only.

# LoginUrl

This object represents a parameter of the inline keyboard button used to automatically authorize a user. Serves as a great replacement for the Telegram Login Widget when the user is coming from Telegram. All the user needs to do is tap/click a button and confirm that they want to log in: Telegram apps support these buttons as of version 5.7.

Sample bot: @discussbot

Source: https://core.telegram.org/bots/api#loginurl

url: str

An HTTPS URL to be opened with user authorization data added to the query string when the button is pressed. If the user refuses to provide authorization data, the original URL without information about the user will be opened. The data added is the same as described in Receiving authorization data.

forward\_text: str | None

Optional. New text of the button in forwarded messages.

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
bot_username: str | None
```

*Optional.* Username of a bot, which will be used for user authorization. See Setting up a bot for more details. If not specified, the current bot's username will be assumed. The *url*'s domain must be the same as the domain linked with the bot. See Linking your domain to the bot for more details.

```
request_write_access: bool | None
```

Optional. Pass True to request the permission for your bot to send messages to the user.

# MaybelnaccessibleMessage

class aiogram.types.maybe\_inaccessible\_message.MaybeInaccessibleMessage(\*\*extra\_data: Any)

This object describes a message that can be inaccessible to the bot. It can be one of

- aiogram.types.message.Message
- aiogram.types.inaccessible\_message.InaccessibleMessage

Source: https://core.telegram.org/bots/api#maybeinaccessiblemessage

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

### **MenuButton**

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

- aiogram.types.menu\_button\_commands.MenuButtonCommands
- aiogram.types.menu\_button\_web\_app.MenuButtonWebApp
- aiogram.types.menu\_button\_default.MenuButtonDefault

If a menu button other than <code>aiogram.types.menu\_button\_default.MenuButtonDefault</code> is set for a private chat, then it is applied in the chat. Otherwise the default menu button is applied. By default, the menu button opens the list of bot commands.

Source: https://core.telegram.org/bots/api#menubutton

type: str

Type of the button

text: str | None

Optional. Text on the button

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
web_app: WebAppInfo | None
```

Optional. Description of the Web App that will be launched when the user presses the button. The Web App will be able to send an arbitrary message on behalf of the user using the method <code>aiogram.methods.answer\_web\_app\_query.AnswerWebAppQuery</code>. Alternatively, a <code>t.me</code> link to a Web App of the bot can be specified in the object instead of the Web App's URL, in which case the Web App will be opened as if the user pressed the link.

#### **MenuButtonCommands**

```
 \textbf{class} \  \, \textbf{aiogram.types.menu\_button\_commands.MenuButtonCommands}(*, type: Literal[MenuButtonType.COMMANDS] \\ = MenuButtonType.COMMANDS, \\ text: str \mid None = None, web\_app: \\ \text{WebAppInfo} \mid None = None, \\ \end{aligned}
```

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

Source: https://core.telegram.org/bots/api#menubuttoncommands

```
type: Literal[MenuButtonType.COMMANDS]
```

Type of the button, must be *commands* 

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

### **MenuButtonDefault**

```
class aiogram.types.menu_button_default.MenuButtonDefault(*, type:
```

```
Literal[MenuButtonType.DEFAULT] =
MenuButtonType.DEFAULT, text: str |
None = None, web_app: WebAppInfo |
None = None, **extra_data: Any)
```

\*\*extra data: Any)

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

Source: https://core.telegram.org/bots/api#menubuttondefault

```
type: Literal[MenuButtonType.DEFAULT]
```

Type of the button, must be default

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model post init method.

# MenuButtonWebApp

class aiogram.types.menu\_button\_web\_app.MenuButtonWebApp(\*, type:

Literal[MenuButtonType.WEB\_APP] = MenuButtonType.WEB\_APP, text: str, web\_app: WebAppInfo, \*\*extra\_data: Any)

Represents a menu button, which launches a Web App.

Source: https://core.telegram.org/bots/api#menubuttonwebapp

type: Literal[MenuButtonType.WEB\_APP]

Type of the button, must be web\_app

text: str

Text on the button

model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

 $model_post_init(context: Any, /) \rightarrow None$ 

We need to both initialize private attributes and call the user-defined model\_post\_init method.

web\_app: WebAppInfo

Description of the Web App that will be launched when the user presses the button. The Web App will be able to send an arbitrary message on behalf of the user using the method <code>aiogram.methods.answer\_web\_app\_query.AnswerWebAppQuery</code>. Alternatively, a t.me link to a Web App of the bot can be specified in the object instead of the Web App's URL, in which case the Web App will be opened as if the user pressed the link.

# Message

**class** alogram.types.message.**Message**(\*, message id: int, date: datetime serializer, return type=int, when\_used=unless - none)], chat: Chat, message\_thread\_id: int | None = None, from user: User | None = None, sender chat: Chat |  $None = None, sender\_boost\_count: int | None = None,$ *sender\_business\_bot:* User | *None = None, business\_connection\_id:* str | None = None, forward origin: MessageOriginUser | MessageOriginHiddenUser | MessageOriginChat | MessageOriginChannel | None = None, is topic message: bool | None = None, is automatic forward: bool | None = None, reply\_to\_message: Message | None = None, external\_reply: ExternalReplyInfo | None = None, quote: TextQuote | None = None, reply\_to\_story: Story | None = None, via\_bot: User | None = None, edit\_date: int | None = None, has\_protected\_content: bool | None = *None*, *is\_from\_offline*: *bool* | *None* = *None*, *media\_group\_id*: *str* | None = None, author\_signature: str | None = None, text: str | None = *None*, *entities*: *List*[MessageEntity] | *None* = *None*, *link\_preview\_options:* LinkPreviewOptions | *None = None, effect\_id:*  $str \mid None = None$ , animation: Animation | None = None, audio: Audio | None = None, document: Document | None = None, paid media: PaidMediaInfo | None = None, photo: List[PhotoSize] | *None* = *None*, *sticker*: Sticker | *None* = *None*, *story*: Story | *None* = None, video: Video | None = None, video note: VideoNote | None = *None*, *voice*: Voice | None = None, caption: str | None = None, caption entities: List[MessageEntity] | None = None, show caption above media:  $bool \mid None = None$ , has media spoiler: bool | None = None, contact: Contact | None = *None, dice:* Dice | None = None, game: Game | None = None, poll: Poll | *None* = *None*, *venue*: Venue | *None* = *None*, *location*: Location  $| None = None, new\_chat\_members: List[User] | None = None,$ *left\_chat\_member:* User | None = None, new\_chat\_title: str | None = *None*,  $new\_chat\_photo$ : List[PhotoSize] | None = None, delete\_chat\_photo: bool | None = None, group\_chat\_created: bool | None = None,  $supergroup\_chat\_created$ :  $bool \mid None = None$ ,  $channel\_chat\_created: bool \mid None = None,$ message auto delete timer changed: MessageAutoDeleteTimerChanged | None = None, migrate to chat id: int | None = None, migrate from chat id: int | *None* = *None*, *pinned\_message*: Message | InaccessibleMessage | *None* = *None*, *invoice*: Invoice | *None* = *None*, *successful* payment: SuccessfulPayment | *None* = *None*, *refunded\_payment*: RefundedPayment | None = None, users shared: UsersShared | None = None, chat shared: ChatShared | None = None, connected website: str | None = None, write access allowed: WriteAccessAllowed |  $None = None, passport\_data: PassportData | None = None,$ *proximity\_alert\_triggered:* ProximityAlertTriggered | *None* = *None*, boost\_added: ChatBoostAdded | None = None, chat\_background\_set: ChatBackground | *None* = *None*, *forum\_topic\_created*: ForumTopicCreated | *None* = *None*, *forum\_topic\_edited*: ForumTopicEdited | *None* = *None*, *forum\_topic\_closed*: ForumTopicClosed | *None* = *None*, *forum\_topic\_reopened*: ForumTopicReopened | *None* = *None*, *general\_forum\_topic\_hidden*: GeneralForumTopicHidden | None = None, general\_forum\_topic\_unhidden: GeneralForumTopicUnhidden | None = None, giveaway created: GiveawayCreated | None = None, *giveaway:* Giveaway | *None* = *None*, *giveaway\_winners:* Giveaway Winners | None = None, giveaway completed: 204 GiveawayCompleted | None = None, video\_c/Chapten/2ed/Contents

> VideoChatScheduled | None = None, video chat started: VideoChatStarted | None = None, video chat ended:

VideoChatEnded | None = None, video chat participants invited:

This object represents a message.

Source: https://core.telegram.org/bots/api#message

# message\_id: int

Unique message identifier inside this chat

# date: DateTime

Date the message was sent in Unix time. It is always a positive number, representing a valid date.

#### chat: Chat

Chat the message belongs to

# message\_thread\_id: int | None

Optional. Unique identifier of a message thread to which the message belongs; for supergroups only

# from\_user: User | None

*Optional*. Sender of the message; may be empty for messages sent to channels. For backward compatibility, if the message was sent on behalf of a chat, the field contains a fake sender user in non-channel chats

### sender\_chat: Chat | None

*Optional*. Sender of the message when sent on behalf of a chat. For example, the supergroup itself for messages sent by its anonymous administrators or a linked channel for messages automatically forwarded to the channel's discussion group. For backward compatibility, if the message was sent on behalf of a chat, the field *from* contains a fake sender user in non-channel chats.

#### sender\_boost\_count: int | None

Optional. If the sender of the message boosted the chat, the number of boosts added by the user

### sender\_business\_bot: User | None

*Optional*. The bot that actually sent the message on behalf of the business account. Available only for outgoing messages sent on behalf of the connected business account.

### business\_connection\_id: str | None

*Optional*. Unique identifier of the business connection from which the message was received. If non-empty, the message belongs to a chat of the corresponding business account that is independent from any potential bot chat which might share the same identifier.

# forward\_origin: MessageOriginUser | MessageOriginHiddenUser | MessageOriginChat | MessageOriginChannel | None

Optional. Information about the original message for forwarded messages

# is\_topic\_message: bool | None

Optional. True, if the message is sent to a forum topic

#### is\_automatic\_forward: bool | None

*Optional*. True, if the message is a channel post that was automatically forwarded to the connected discussion group

# reply\_to\_message: Message | None

*Optional*. For replies in the same chat and message thread, the original message. Note that the Message object in this field will not contain further *reply\_to\_message* fields even if it itself is a reply.

#### external\_reply: ExternalReplyInfo | None

*Optional*. Information about the message that is being replied to, which may come from another chat or forum topic

quote: TextQuote | None Optional. For replies that quote part of the original message, the quoted part of the message reply\_to\_story: Story | None Optional. For replies to a story, the original story via\_bot: User | None Optional. Bot through which the message was sent edit\_date: int | None Optional. Date the message was last edited in Unix time has\_protected\_content: bool | None Optional. True, if the message can't be forwarded is\_from\_offline: bool | None Optional. True, if the message was sent by an implicit action, for example, as an away or a greeting business message, or as a scheduled message media\_group\_id: str | None Optional. The unique identifier of a media message group this message belongs to author\_signature: str | None Optional. Signature of the post author for messages in channels, or the custom title of an anonymous group administrator text: str | None Optional. For text messages, the actual UTF-8 text of the message entities: List[MessageEntity] | None Optional. For text messages, special entities like usernames, URLs, bot commands, etc. that appear in the text link\_preview\_options: LinkPreviewOptions | None Optional. Options used for link preview generation for the message, if it is a text message and link preview options were changed effect\_id: str | None Optional. Unique identifier of the message effect added to the message animation: Animation | None Optional. Message is an animation, information about the animation. For backward compatibility, when this field is set, the document field will also be set audio: Audio | None Optional. Message is an audio file, information about the file document: Document | None Optional. Message is a general file, information about the file paid\_media: PaidMediaInfo | None Optional. Message contains paid media; information about the paid media photo: List[PhotoSize] | None Optional. Message is a photo, available sizes of the photo sticker: Sticker | None

Optional. Message is a sticker, information about the sticker

```
story: Story | None
     Optional. Message is a forwarded story
video: Video | None
     Optional. Message is a video, information about the video
video_note: VideoNote | None
     Optional. Message is a video note, information about the video message
voice: Voice | None
     Optional. Message is a voice message, information about the file
caption: str | None
     Optional. Caption for the animation, audio, document, paid media, photo, video or voice
caption_entities: List[MessageEntity] | None
     Optional. For messages with a caption, special entities like usernames, URLs, bot commands, etc. that
     appear in the caption
show_caption_above_media: bool | None
     Optional. True, if the caption must be shown above the message media
has_media_spoiler: bool | None
     Optional. True, if the message media is covered by a spoiler animation
contact: Contact | None
     Optional. Message is a shared contact, information about the contact
dice: Dice | None
     Optional. Message is a dice with random value
game: Game | None
     Optional. Message is a game, information about the game. More about games »
poll: Poll | None
     Optional. Message is a native poll, information about the poll
venue: Venue | None
     Optional. Message is a venue, information about the venue. For backward compatibility, when this field is
     set, the location field will also be set
location: Location | None
     Optional. Message is a shared location, information about the location
new_chat_members: List[User] | None
     Optional. New members that were added to the group or supergroup and information about them (the bot
     itself may be one of these members)
left_chat_member: User | None
     Optional. A member was removed from the group, information about them (this member may be the bot
     itself)
new_chat_title: str | None
     Optional. A chat title was changed to this value
new_chat_photo: List[PhotoSize] | None
     Optional. A chat photo was change to this value
```

delete\_chat\_photo: bool | None

Optional. Service message: the chat photo was deleted

group\_chat\_created: bool | None

Optional. Service message: the group has been created

supergroup\_chat\_created: bool | None

*Optional*. Service message: the supergroup has been created. This field can't be received in a message coming through updates, because bot can't be a member of a supergroup when it is created. It can only be found in reply\_to\_message if someone replies to a very first message in a directly created supergroup.

channel\_chat\_created: bool | None

*Optional*. Service message: the channel has been created. This field can't be received in a message coming through updates, because bot can't be a member of a channel when it is created. It can only be found in reply\_to\_message if someone replies to a very first message in a channel.

message\_auto\_delete\_timer\_changed: MessageAutoDeleteTimerChanged | None

Optional. Service message: auto-delete timer settings changed in the chat

migrate\_to\_chat\_id: int | None

Optional. The group has been migrated to a supergroup with the specified identifier. This number may have more than 32 significant bits and some programming languages may have difficulty/silent defects in interpreting it. But it has at most 52 significant bits, so a signed 64-bit integer or double-precision float type are safe for storing this identifier.

migrate\_from\_chat\_id: int | None

*Optional.* The supergroup has been migrated from a group with the specified identifier. This number may have more than 32 significant bits and some programming languages may have difficulty/silent defects in interpreting it. But it has at most 52 significant bits, so a signed 64-bit integer or double-precision float type are safe for storing this identifier.

pinned\_message: Message | InaccessibleMessage | None

*Optional.* Specified message was pinned. Note that the Message object in this field will not contain further *reply\_to\_message* fields even if it itself is a reply.

invoice: *Invoice* | None

Optional. Message is an invoice for a payment, information about the invoice. More about payments »

successful\_payment: SuccessfulPayment | None

*Optional*. Message is a service message about a successful payment, information about the payment. More about payments »

refunded\_payment: RefundedPayment | None

*Optional*. Message is a service message about a refunded payment, information about the payment. More about payments »

users\_shared: UsersShared | None

Optional. Service message: users were shared with the bot

chat\_shared: ChatShared | None

Optional. Service message: a chat was shared with the bot

connected\_website: str | None

Optional. The domain name of the website on which the user has logged in. More about Telegram Login »

```
write_access_allowed: WriteAccessAllowed | None
    Optional. Service message: the user allowed the bot to write messages after adding it to the attachment or
    side menu, launching a Web App from a link, or accepting an explicit request from a Web App sent by the
    method requestWriteAccess
passport_data: PassportData | None
    Optional. Telegram Passport data
proximity_alert_triggered: ProximityAlertTriggered | None
    Optional. Service message. A user in the chat triggered another user's proximity alert while sharing Live
    Location.
boost_added: ChatBoostAdded | None
    Optional. Service message: user boosted the chat
chat_background_set: ChatBackground | None
    Optional. Service message: chat background set
forum_topic_created: ForumTopicCreated | None
    Optional. Service message: forum topic created
forum_topic_edited: ForumTopicEdited | None
    Optional. Service message: forum topic edited
forum_topic_closed: ForumTopicClosed | None
    Optional. Service message: forum topic closed
forum_topic_reopened: ForumTopicReopened | None
    Optional. Service message: forum topic reopened
general_forum_topic_hidden: GeneralForumTopicHidden | None
    Optional. Service message: the 'General' forum topic hidden
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
    A dictionary of computed field names and their corresponding ComputedFieldInfo objects.
model_post_init(context: Any, /) \rightarrow None
    We need to both initialize private attributes and call the user-defined model post init method.
general_forum_topic_unhidden: GeneralForumTopicUnhidden | None
    Optional. Service message: the 'General' forum topic unhidden
giveaway_created: GiveawayCreated | None
    Optional. Service message: a scheduled giveaway was created
giveaway: Giveaway | None
    Optional. The message is a scheduled giveaway message
giveaway_winners: GiveawayWinners | None
    Optional. A giveaway with public winners was completed
giveaway_completed: GiveawayCompleted | None
    Optional. Service message: a giveaway without public winners was completed
video_chat_scheduled: VideoChatScheduled | None
    Optional. Service message: video chat scheduled
```

```
video_chat_started: VideoChatStarted | None
     Optional. Service message: video chat started
video_chat_ended: VideoChatEnded | None
     Optional. Service message: video chat ended
video_chat_participants_invited: VideoChatParticipantsInvited | None
     Optional. Service message: new participants invited to a video chat
web_app_data: WebAppData | None
     Optional. Service message: data sent by a Web App
reply_markup: InlineKeyboardMarkup | None
     Optional. Inline keyboard attached to the message. login_url buttons are represented as ordinary url
     buttons.
forward_date: DateTime | None
     Optional. For forwarded messages, date the original message was sent in Unix time
     Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023
forward_from: User | None
     Optional. For forwarded messages, sender of the original message
     Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023
forward_from_chat: Chat | None
     Optional. For messages forwarded from channels or from anonymous administrators, information about
     the original sender chat
     Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023
forward_from_message_id: int | None
     Optional. For messages forwarded from channels, identifier of the original message in the channel
     Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023
forward_sender_name: str | None
     Optional. Sender's name for messages forwarded from users who disallow adding a link to their account
     in forwarded messages
     Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023
forward_signature: str | None
     Optional. For forwarded messages that were originally sent in channels or by an anonymous chat adminis-
     trator, signature of the message sender if present
     Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023
user_shared: UserShared | None
     Optional. Service message: a user was shared with the bot
     Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023
property content_type: str
property html_text: str
property md_text: str
```

```
as_reply_parameters(allow\_sending\_without\_reply: Optional[Union[bool, Default]] = \\ < Default('allow\_sending\_without\_reply')>, quote: <math>Optional[str] = None, \\ quote\_parse\_mode: Optional[Union[str, Default]] = < Default('parse\_mode')>, \\ quote\_entities: Optional[List[MessageEntity]] = None, quote\_position: \\ Optional[int] = None) \rightarrow ReplyParameters
```

Shortcut for method aiogram.methods.send\_animation.SendAnimation will automatically fill method attributes:

- chat\_id
- message\_thread\_id
- business\_connection\_id
- reply\_parameters

Use this method to send animation files (GIF or H.264/MPEG-4 AVC video without sound). On success, the sent <code>aiogram.types.message.Message</code> 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 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. More information on Sending Files »
- duration Duration of sent animation in seconds
- width Animation width
- height Animation height
- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file\_attach\_name>' if the thumbnail was uploaded using multipart/form-data under <file\_attach\_name>. *More information on Sending Files*
- **caption** Animation caption (may also be used when resending animation by *file\_id*), 0-1024 characters after entities parsing
- parse\_mode Mode for parsing entities in the animation caption. See formatting options for more details.

- **caption\_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse\_mode*
- **show\_caption\_above\_media** Pass True, if the caption must be shown above the message media
- has\_spoiler Pass True if the animation needs to be covered with a spoiler animation
- disable\_notification Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found

instance of method aiogram.methods.send\_animation.SendAnimation

```
answer_animation (animation: Union[InputFile, str], duration: Optional[int] = None, width: Optional[int] = None, height: Optional[int] = None, thumbnail: Optional[InputFile] = None, caption: Optional[str] = None, parse_mode: Optional[Union[str, Default]] = <Default('parse_mode')>, caption_entities: Optional[List[MessageEntity]] = None, show_caption_above_media: Optional[Union[bool, Default]] = <Default('show_caption_above_media')>, has_spoiler: Optional[bool] = None, disable_notification: Optional[bool] = None, protect_content: Optional[Union[bool, Default]] = <Default('protect_content')>, message_effect_id: Optional[str] = None, reply_parameters: Optional[ReplyParameters] = None, reply_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None, allow_sending_without_reply: Optional[bool] = None, reply_to_message_id: Optional[int] = None, **kwargs: Any) → SendAnimation
```

Shortcut for method  $aiogram.methods.send\_animation.SendAnimation$  will automatically fill method attributes:

- chat\_id
- message\_thread\_id
- business\_connection\_id

Use this method to send animation files (GIF or H.264/MPEG-4 AVC video without sound). On success, the sent aiogram.types.message.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

- animation 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. *More information on Sending Files* »
- duration Duration of sent animation in seconds

- width Animation width
- **height** Animation height
- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file\_attach\_name>' if the thumbnail was uploaded using multipart/form-data under <file\_attach\_name>. *More information on Sending Files*
- **caption** Animation caption (may also be used when resending animation by *file\_id*), 0-1024 characters after entities parsing
- parse\_mode Mode for parsing entities in the animation caption. See formatting options for more details.
- **caption\_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse\_mode*
- **show\_caption\_above\_media** Pass True, if the caption must be shown above the message media
- has\_spoiler Pass True if the animation needs to be covered with a spoiler animation
- disable\_notification Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send\_animation.SendAnimation

reply\_audio(audio: Union[InputFile, str], caption: Optional[str] = None, parse\_mode: Optional[Union[str, Default]] = <Default('parse\_mode')>, caption\_entities: Optional[List[MessageEntity]] = None, duration: Optional[int] = None, performer: Optional[str] = None, title: Optional[str] = None, thumbnail: Optional[InputFile] = None, disable\_notification: Optional[bool] = None, protect\_content: Optional[Union[bool, Default]] = <Default('protect\_content')>, message\_effect\_id: Optional[str] = None, reply\_markup:
Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None, allow\_sending\_without\_reply: Optional[bool] = None, \*\*kwargs: Any)
→ SendAudio

Shortcut for method aiogram.methods.send\_audio.SendAudio will automatically fill method attributes:

- chat\_id
- message\_thread\_id

- business\_connection\_id
- reply\_parameters

Use this method to send audio files, if you want Telegram clients to display them in the music player. Your audio must be in the .MP3 or .M4A format. On success, the sent <code>aiogram.types.message.Message</code> is returned. Bots can currently send audio files of up to 50 MB in size, this limit may be changed in the future. For sending voice messages, use the <code>aiogram.methods.send\_voice.SendVoice</code> method instead.

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

### **Parameters**

- audio Audio file to send. Pass a file\_id as String to send an audio file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get an audio file from the Internet, or upload a new one using multipart/form-data. *More information on Sending Files* »
- caption Audio caption, 0-1024 characters after entities parsing
- parse\_mode Mode for parsing entities in the audio caption. See formatting options for more details.
- **caption\_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse\_mode*
- duration Duration of the audio in seconds
- **performer** Performer
- title Track name
- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file\_attach\_name>' if the thumbnail was uploaded using multipart/form-data under <file\_attach\_name>. *More information on Sending Files*""
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found

### Returns

instance of method aiogram.methods.send\_audio.SendAudio

answer\_audio(audio: Union[InputFile, str], caption: Optional[str] = None, parse\_mode:

Optional[Union[str, Default]] = <Default('parse\_mode')>, caption\_entities:

Optional[List[MessageEntity]] = None, duration: Optional[int] = None, performer:

Optional[str] = None, title: Optional[str] = None, thumbnail: Optional[InputFile] = None, disable\_notification: Optional[bool] = None, protect\_content: Optional[Union[bool, Default]] = <Default('protect\_content')>, message\_effect\_id: Optional[str] = None, reply\_parameters: Optional[ReplyParameters] = None, reply\_markup:

Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None, allow\_sending\_without\_reply: Optional[bool] = None, reply\_to\_message\_id: Optional[int] = None, \*\*kwargs: Any) → SendAudio

Shortcut for method aiogram.methods.send\_audio.SendAudio will automatically fill method attributes:

- chat\_id
- message\_thread\_id
- business\_connection\_id

Use this method to send audio files, if you want Telegram clients to display them in the music player. Your audio must be in the .MP3 or .M4A format. On success, the sent <code>aiogram.types.message.Message</code> is returned. Bots can currently send audio files of up to 50 MB in size, this limit may be changed in the future. For sending voice messages, use the <code>aiogram.methods.send\_voice.SendVoice</code> method instead.

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

#### **Parameters**

- audio Audio file to send. Pass a file\_id as String to send an audio file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get an audio file from the Internet, or upload a new one using multipart/form-data. *More information on Sending Files* »
- caption Audio caption, 0-1024 characters after entities parsing
- parse\_mode Mode for parsing entities in the audio caption. See formatting options for more details.
- **caption\_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse\_mode*
- duration Duration of the audio in seconds
- **performer** Performer
- title Track name
- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file\_attach\_name>' if the thumbnail was uploaded using multipart/form-data under <file\_attach\_name>. *More information on Sending Files*»
- disable\_notification Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only

- **reply\_parameters** Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send\_audio.SendAudio

```
reply_contact(phone_number: str, first_name: str, last_name: Optional[str] = None, vcard: Optional[str] = None, disable_notification: Optional[bool] = None, protect_content:

Optional[Union[bool, Default]] = <Default('protect_content')>, message_effect_id:

Optional[str] = None, reply_markup: Optional[Union[InlineKeyboardMarkup,

ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None,

allow_sending_without_reply: Optional[bool] = None, **kwargs: Any) → SendContact
```

Shortcut for method <code>aiogram.methods.send\_contact.SendContact</code> will automatically fill method attributes:

- chat\_id
- message\_thread\_id
- business\_connection\_id
- reply\_parameters

Use this method to send phone contacts. On success, the sent aiogram.types.message.Message is returned.

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

## **Parameters**

- **phone\_number** Contact's phone number
- first\_name Contact's first name
- last\_name Contact's last name
- vcard Additional data about the contact in the form of a vCard, 0-2048 bytes
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found

## Returns

instance of method aiogram.methods.send\_contact.SendContact

```
answer_contact(phone_number: str, first_name: str, last_name: Optional[str] = None, vcard:

Optional[str] = None, disable_notification: Optional[bool] = None, protect_content:

Optional[Union[bool, Default]] = <Default('protect_content')>, message_effect_id:

Optional[str] = None, reply_parameters: Optional[ReplyParameters] = None,

reply_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup,

ReplyKeyboardRemove, ForceReply]] = None, allow_sending_without_reply:

Optional[bool] = None, reply_to_message_id: Optional[int] = None, **kwargs: Any) →

SendContact
```

Shortcut for method <code>aiogram.methods.send\_contact.SendContact</code> will automatically fill method attributes:

- chat\_id
- message\_thread\_id
- business\_connection\_id

Use this method to send phone contacts. On success, the sent aiogram.types.message.Message is returned.

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

## **Parameters**

- **phone\_number** Contact's phone number
- first\_name Contact's first name
- last\_name Contact's last name
- vcard Additional data about the contact in the form of a vCard, 0-2048 bytes
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

### Returns

instance of method aiogram.methods.send\_contact.SendContact

```
reply_document (document: Union[InputFile, str], thumbnail: Optional[InputFile] = None, caption:

Optional[str] = None, parse_mode: Optional[Union[str, Default]] =

<Default('parse_mode')>, caption_entities: Optional[List[MessageEntity]] = None,
disable_content_type_detection: Optional[bool] = None, disable_notification:
Optional[bool] = None, protect_content: Optional[Union[bool, Default]] =

<Default('protect_content')>, message_effect_id: Optional[str] = None, reply_markup:
Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove,
ForceReply]] = None, allow_sending_without_reply: Optional[bool] = None, **kwargs:
Any) → SendDocument
```

Shortcut for method aiogram.methods.send\_document.SendDocument will automatically fill method attributes:

- chat\_id
- message\_thread\_id
- business\_connection\_id
- reply\_parameters

Use this method to send general files. On success, the sent aiogram.types.message.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** File to send. Pass a file\_id as String to send a file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a file from the Internet, or upload a new one using multipart/form-data. *More information on Sending Files* »
- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file\_attach\_name>' if the thumbnail was uploaded using multipart/form-data under <file\_attach\_name>. *More information on Sending Files*
- **caption** Document caption (may also be used when resending documents by *file\_id*), 0-1024 characters after entities parsing
- parse\_mode Mode for parsing entities in the document caption. See formatting options for more details.
- **caption\_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse\_mode*
- **disable\_content\_type\_detection** Disables automatic server-side content type detection for files uploaded using multipart/form-data
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found

# Returns

instance of method aiogram.methods.send\_document.SendDocument

```
answer_document: Union[InputFile, str], thumbnail: Optional[InputFile] = None, caption:

Optional[str] = None, parse_mode: Optional[Union[str, Default]] =

<Default('parse_mode')>, caption_entities: Optional[List[MessageEntity]] = None,
disable_content_type_detection: Optional[bool] = None, disable_notification:
Optional[bool] = None, protect_content: Optional[Union[bool, Default]] =

<Default('protect_content')>, message_effect_id: Optional[str] = None,
reply_parameters: Optional[ReplyParameters] = None, reply_markup:
Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove,
ForceReply]] = None, allow_sending_without_reply: Optional[bool] = None,
reply_to_message_id: Optional[int] = None, **kwargs: Any) → SendDocument
```

Shortcut for method <code>aiogram.methods.send\_document.SendDocument</code> will automatically fill method attributes:

- chat\_id
- message\_thread\_id
- business\_connection\_id

Use this method to send general files. On success, the sent aiogram.types.message.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** File to send. Pass a file\_id as String to send a file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a file from the Internet, or upload a new one using multipart/form-data. *More information on Sending Files* »
- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file\_attach\_name>' if the thumbnail was uploaded using multipart/form-data under <file\_attach\_name>. *More information on Sending Files*
- **caption** Document caption (may also be used when resending documents by *file\_id*), 0-1024 characters after entities parsing
- parse\_mode Mode for parsing entities in the document caption. See formatting options for more details.
- **caption\_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse\_mode*
- **disable\_content\_type\_detection** Disables automatic server-side content type detection for files uploaded using multipart/form-data
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- **message\_effect\_id** Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to

- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send\_document.SendDocument

 $\textbf{reply\_game}(game\_short\_name: str, disable\_notification: Optional[bool] = None, protect\_content: \\ Optional[Union[bool, Default]] = < Default('protect\_content') >, message\_effect\_id: \\ Optional[str] = None, reply\_markup: Optional[InlineKeyboardMarkup] = None, \\ allow\_sending\_without\_reply: Optional[bool] = None, **kwargs: Any) \rightarrow SendGame \\$ 

Shortcut for method aiogram.methods.send\_game.SendGame will automatically fill method attributes:

- chat\_id
- message\_thread\_id
- business\_connection\_id
- reply\_parameters

Use this method to send a game. On success, the sent aiogram.types.message.Message is returned.

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

#### **Parameters**

- game\_short\_name Short name of the game, serves as the unique identifier for the game. Set up your games via @BotFather.
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- **message\_effect\_id** Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_markup** 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.
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found

### Returns

instance of method aiogram.methods.send\_game.SendGame

 $\begin{tabular}{ll} \textbf{answer\_game} (game\_short\_name: str, disable\_notification: Optional[bool] = None, protect\_content: \\ Optional[Union[bool, Default]] = & Default('protect\_content') >, message\_effect\_id: \\ Optional[str] = None, reply\_parameters: Optional[ReplyParameters] = None, reply\_markup: \\ Optional[InlineKeyboardMarkup] = None, allow\_sending\_without\_reply: Optional[bool] = \\ None, reply\_to\_message\_id: Optional[int] = None, **kwargs: Any) \rightarrow SendGame \\ \end{tabular}$ 

Shortcut for method <code>aiogram.methods.send\_game.SendGame</code> will automatically fill method attributes:

- chat\_id
- message\_thread\_id
- business\_connection\_id

Use this method to send a game. On success, the sent aiogram.types.message.Message is returned.

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

## **Parameters**

- game\_short\_name Short name of the game, serves as the unique identifier for the game. Set up your games via @BotFather.
- disable\_notification Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** 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.
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

### Returns

instance of method aiogram.methods.send\_game.SendGame

Shortcut for method <code>aiogram.methods.send\_invoice.SendInvoice</code> will automatically fill method attributes:

- chat id
- message\_thread\_id
- business\_connection\_id
- reply\_parameters

Use this method to send invoices. On success, the sent aiogram.types.message.Message is returned.

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

### **Parameters**

- title Product name, 1-32 characters
- **description** Product description, 1-255 characters

- payload Bot-defined invoice payload, 1-128 bytes. This will not be displayed to the user, use it for your internal processes.
- **currency** Three-letter ISO 4217 currency code, see more on currencies. Pass 'XTR' for payments in Telegram Stars.
- **prices** Price breakdown, a JSON-serialized list of components (e.g. product price, tax, discount, delivery cost, delivery tax, bonus, etc.). Must contain exactly one item for payments in Telegram Stars.
- **provider\_token** Payment provider token, obtained via @BotFather. Pass an empty string for payments in Telegram Stars.
- max\_tip\_amount The maximum accepted amount for tips in the *smallest units* of the currency (integer, **not** float/double). For example, for a maximum tip of US\$ 1.45 pass max\_tip\_amount = 145. See the *exp* parameter in currencies.json, it shows the number of digits past the decimal point for each currency (2 for the majority of currencies). Defaults to 0. Not supported for payments in Telegram Stars.
- **suggested\_tip\_amounts** A JSON-serialized array of suggested amounts of tips in the *smallest units* of the currency (integer, **not** float/double). At most 4 suggested tip amounts can be specified. The suggested tip amounts must be positive, passed in a strictly increased order and must not exceed *max\_tip\_amount*.
- **start\_parameter** Unique deep-linking parameter. If left empty, **forwarded copies** of the sent message will have a *Pay* button, allowing multiple users to pay directly from the forwarded message, using the same invoice. If non-empty, forwarded copies of the sent message will have a *URL* button with a deep link to the bot (instead of a *Pay* button), with the value used as the start parameter
- **provider\_data** JSON-serialized data about the invoice, which will be shared with the payment provider. A detailed description of required fields should be provided by the payment provider.
- **photo\_url** URL of the product photo for the invoice. Can be a photo of the goods or a marketing image for a service. People like it better when they see what they are paying for.
- photo\_size Photo size in bytes
- **photo\_width** Photo width
- photo\_height Photo height
- **need\_name** Pass True if you require the user's full name to complete the order. Ignored for payments in Telegram Stars.
- need\_phone\_number Pass True if you require the user's phone number to complete the order. Ignored for payments in Telegram Stars.
- need\_email Pass True if you require the user's email address to complete the order. Ignored for payments in Telegram Stars.
- **need\_shipping\_address** Pass True if you require the user's shipping address to complete the order. Ignored for payments in Telegram Stars.
- **send\_phone\_number\_to\_provider** Pass True if the user's phone number should be sent to the provider. Ignored for payments in Telegram Stars.
- **send\_email\_to\_provider** Pass True if the user's email address should be sent to the provider. Ignored for payments in Telegram Stars.
- **is\_flexible** Pass True if the final price depends on the shipping method. Ignored for payments in Telegram Stars.

- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_markup** 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.
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found

instance of method aiogram.methods.send\_invoice.SendInvoice

Shortcut for method  $aiogram.methods.send\_invoice.SendInvoice$  will automatically fill method attributes:

- chat id
- message\_thread\_id
- business\_connection\_id

Use this method to send invoices. On success, the sent aiogram.types.message.Message is returned.

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

## **Parameters**

- title Product name, 1-32 characters
- **description** Product description, 1-255 characters
- **payload** Bot-defined invoice payload, 1-128 bytes. This will not be displayed to the user, use it for your internal processes.
- **currency** Three-letter ISO 4217 currency code, see more on currencies. Pass 'XTR' for payments in Telegram Stars.
- **prices** Price breakdown, a JSON-serialized list of components (e.g. product price, tax, discount, delivery cost, delivery tax, bonus, etc.). Must contain exactly one item for payments in Telegram Stars.
- **provider\_token** Payment provider token, obtained via @BotFather. Pass an empty string for payments in Telegram Stars.

- max\_tip\_amount The maximum accepted amount for tips in the *smallest units* of the currency (integer, **not** float/double). For example, for a maximum tip of US\$ 1.45 pass max\_tip\_amount = 145. See the *exp* parameter in currencies.json, it shows the number of digits past the decimal point for each currency (2 for the majority of currencies). Defaults to 0. Not supported for payments in Telegram Stars.
- **suggested\_tip\_amounts** A JSON-serialized array of suggested amounts of tips in the *smallest units* of the currency (integer, **not** float/double). At most 4 suggested tip amounts can be specified. The suggested tip amounts must be positive, passed in a strictly increased order and must not exceed *max\_tip\_amount*.
- **start\_parameter** Unique deep-linking parameter. If left empty, **forwarded copies** of the sent message will have a *Pay* button, allowing multiple users to pay directly from the forwarded message, using the same invoice. If non-empty, forwarded copies of the sent message will have a *URL* button with a deep link to the bot (instead of a *Pay* button), with the value used as the start parameter
- **provider\_data** JSON-serialized data about the invoice, which will be shared with the payment provider. A detailed description of required fields should be provided by the payment provider.
- **photo\_url** URL of the product photo for the invoice. Can be a photo of the goods or a marketing image for a service. People like it better when they see what they are paying for.
- **photo\_size** Photo size in bytes
- photo\_width Photo width
- photo\_height Photo height
- **need\_name** Pass True if you require the user's full name to complete the order. Ignored for payments in Telegram Stars.
- **need\_phone\_number** Pass True if you require the user's phone number to complete the order. Ignored for payments in Telegram Stars.
- **need\_email** Pass True if you require the user's email address to complete the order. Ignored for payments in Telegram Stars.
- **need\_shipping\_address** Pass True if you require the user's shipping address to complete the order. Ignored for payments in Telegram Stars.
- **send\_phone\_number\_to\_provider** Pass True if the user's phone number should be sent to the provider. Ignored for payments in Telegram Stars.
- **send\_email\_to\_provider** Pass True if the user's email address should be sent to the provider. Ignored for payments in Telegram Stars.
- **is\_flexible** Pass True if the final price depends on the shipping method. Ignored for payments in Telegram Stars.
- disable\_notification Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** 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.

- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send\_invoice.SendInvoice

```
reply_location(latitude: float, longitude: float, horizontal_accuracy: Optional[float] = None, live_period:

Optional[int] = None, heading: Optional[int] = None, proximity_alert_radius:

Optional[int] = None, disable_notification: Optional[bool] = None, protect_content:

Optional[Union[bool, Default]] = <Default('protect_content')>, message_effect_id:

Optional[str] = None, reply_markup: Optional[Union[InlineKeyboardMarkup,

ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None,

allow_sending_without_reply: Optional[bool] = None, **kwargs: Any) → SendLocation
```

Shortcut for method <code>aiogram.methods.send\_location.SendLocation</code> will automatically fill method attributes:

- chat\_id
- message\_thread\_id
- business\_connection\_id
- reply\_parameters

Use this method to send point on the map. On success, the sent aiogram.types.message.Message is returned.

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

### **Parameters**

- latitude Latitude of the location
- longitude Longitude of the location
- horizontal\_accuracy The radius of uncertainty for the location, measured in meters;
   0-1500
- **live\_period** Period in seconds during which the location will be updated (see Live Locations, should be between 60 and 86400, or 0x7FFFFFFF for live locations that can be edited indefinitely.
- **heading** For live locations, a direction in which the user is moving, in degrees. Must be between 1 and 360 if specified.
- **proximity\_alert\_radius** 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** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- **message\_effect\_id** Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

• allow\_sending\_without\_reply - Pass True if the message should be sent even if the specified replied-to message is not found

#### Returns

instance of method aiogram.methods.send\_location.SendLocation

```
answer_location(latitude: float, longitude: float, horizontal_accuracy: Optional[float] = None, live_period: Optional[int] = None, heading: Optional[int] = None, proximity_alert_radius: Optional[int] = None, disable_notification: Optional[bool] = None, protect_content: Optional[Union[bool, Default]] = <Default('protect_content')>, message_effect_id: Optional[str] = None, reply_parameters: Optional[ReplyParameters] = None, reply_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None, allow_sending_without_reply: Optional[bool] = None, reply_to_message_id: Optional[int] = None, **kwargs: Any) → SendLocation
```

Shortcut for method aiogram.methods.send\_location.SendLocation will automatically fill method attributes:

- chat\_id
- message\_thread\_id
- business\_connection\_id

Use this method to send point on the map. On success, the sent aiogram.types.message.Message is returned.

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

- latitude Latitude of the location
- longitude Longitude of the location
- horizontal\_accuracy The radius of uncertainty for the location, measured in meters; 0-1500
- **live\_period** Period in seconds during which the location will be updated (see Live Locations, should be between 60 and 86400, or 0x7FFFFFFF for live locations that can be edited indefinitely.
- **heading** For live locations, a direction in which the user is moving, in degrees. Must be between 1 and 360 if specified.
- **proximity\_alert\_radius** 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** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send\_location.SendLocation

 $\textbf{reply\_media\_group}(media: List[Union[InputMediaAudio, InputMediaDocument, InputMediaPhoto, \\ InputMediaVideo]], disable\_notification: Optional[bool] = None, protect\_content: \\ Optional[Union[bool, Default]] = <Default('protect\_content')>, message\_effect\_id: \\ Optional[str] = None, allow\_sending\_without\_reply: Optional[bool] = None, \\ **kwargs: Any) \rightarrow SendMediaGroup$ 

Shortcut for method  $aiogram.methods.send\_media\_group.SendMediaGroup$  will automatically fill method attributes:

- chat id
- message\_thread\_id
- business\_connection\_id
- reply\_parameters

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

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

#### **Parameters**

- media A JSON-serialized array describing messages to be sent, must include 2-10 items
- disable\_notification Sends messages silently. Users will receive a notification with no sound.
- **protect\_content** Protects the contents of the sent messages from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found

## Returns

instance of method aiogram.methods.send\_media\_group.SendMediaGroup

answer\_media\_group(media: List[Union[InputMediaAudio, InputMediaDocument, InputMediaPhoto, InputMediaVideo]], disable\_notification: Optional[bool] = None, protect\_content: Optional[Union[bool, Default]] =  $\langle Default('protect\_content') \rangle$ , message\_effect\_id: Optional[str] = None, reply\_parameters: Optional[ReplyParameters] = None, allow\_sending\_without\_reply: Optional[bool] = None, reply\_to\_message\_id: Optional[int] = None, \*\*kwargs: Any)  $\rightarrow$  SendMediaGroup

Shortcut for method  $aiogram.methods.send\_media\_group.SendMediaGroup$  will automatically fill method attributes:

- chat\_id
- message\_thread\_id
- business\_connection\_id

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

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

## **Parameters**

- media A JSON-serialized array describing messages to be sent, must include 2-10 items
- disable\_notification Sends messages silently. Users will receive a notification with no sound.
- **protect\_content** Protects the contents of the sent messages from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the messages are a reply, ID of the original message

## Returns

instance of method aiogram.methods.send\_media\_group.SendMediaGroup

reply(text: str, parse\_mode: Optional[Union[str, Default]] = <Default('parse\_mode')>, entities:

Optional[List[MessageEntity]] = None, link\_preview\_options: Optional[Union[LinkPreviewOptions,
Default]] = <Default('link\_preview')>, disable\_notification: Optional[bool] = None, protect\_content:
Optional[Union[bool, Default]] = <Default('protect\_content')>, message\_effect\_id: Optional[str] =
None, reply\_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup,
ReplyKeyboardRemove, ForceReply]] = None, allow\_sending\_without\_reply: Optional[bool] = None,
disable\_web\_page\_preview: Optional[Union[bool, Default]] =
<Default('link\_preview\_is\_disabled')>, \*\*kwargs: Any) → SendMessage

Shortcut for method  $aiogram.methods.send\_message.SendMessage$  will automatically fill method attributes:

- chat\_id
- message\_thread\_id
- business\_connection\_id
- reply\_parameters

Use this method to send text messages. On success, the sent aiogram.types.message.Message is returned.

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

- text Text of the message to be sent, 1-4096 characters after entities parsing
- parse\_mode Mode for parsing entities in the message text. See formatting options for more details.
- **entities** A JSON-serialized list of special entities that appear in message text, which can be specified instead of *parse\_mode*
- link\_preview\_options Link preview generation options for the message

- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- disable\_web\_page\_preview Disables link previews for links in this message

instance of method aiogram.methods.send\_message.SendMessage

```
answer(text: str, parse_mode: Optional[Union[str, Default]] = <Default('parse_mode')>, entities:

Optional[List[MessageEntity]] = None, link_preview_options: Optional[Union[LinkPreviewOptions,
Default]] = <Default('link_preview')>, disable_notification: Optional[bool] = None,
protect_content: Optional[Union[bool, Default]] = <Default('protect_content')>, message_effect_id:
Optional[str] = None, reply_parameters: Optional[ReplyParameters] = None, reply_markup:
Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove,
ForceReply]] = None, allow_sending_without_reply: Optional[bool] = None,
disable_web_page_preview: Optional[Union[bool, Default]] =
<Default('link_preview_is_disabled')>, reply_to_message_id: Optional[int] = None, **kwargs:
Any) → SendMessage
```

Shortcut for method <code>aiogram.methods.send\_message.SendMessage</code> will automatically fill method attributes:

- chat\_id
- message\_thread\_id
- business\_connection\_id

Use this method to send text messages. On success, the sent aiogram.types.message.Message is returned.

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

## **Parameters**

- text Text of the message to be sent, 1-4096 characters after entities parsing
- parse\_mode Mode for parsing entities in the message text. See formatting options for more details.
- **entities** A JSON-serialized list of special entities that appear in message text, which can be specified instead of *parse\_mode*
- link\_preview\_options Link preview generation options for the message
- disable\_notification Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- **message\_effect\_id** Unique identifier of the message effect to be added to the message; for private chats only

- **reply\_parameters** Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- disable\_web\_page\_preview Disables link previews for links in this message
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send\_message.SendMessage

reply\_photo(photo: Union[InputFile, str], caption: Optional[str] = None, parse\_mode: Optional[Union[str, Default]] = <Default('parse\_mode')>, caption\_entities: Optional[List[MessageEntity]] = None, show\_caption\_above\_media: Optional[Union[bool, Default]] = <Default('show\_caption\_above\_media')>, has\_spoiler: Optional[bool] = None, disable\_notification: Optional[bool] = None, protect\_content: Optional[Union[bool, Default]] = <Default('protect\_content')>, message\_effect\_id: Optional[str] = None, reply\_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None, allow\_sending\_without\_reply: Optional[bool] = None, \*\*kwargs: Any) → SendPhoto

Shortcut for method aiogram.methods.send\_photo.SendPhoto will automatically fill method attributes:

- chat\_id
- message\_thread\_id
- business\_connection\_id
- reply\_parameters

Use this method to send photos. On success, the sent aiogram.types.message.Message is returned.

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

- **photo** Photo to send. Pass a file\_id as String to send a photo that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a photo from the Internet, or upload a new photo using multipart/form-data. The photo must be at most 10 MB in size. The photo's width and height must not exceed 10000 in total. Width and height ratio must be at most 20. *More information on Sending Files* »
- **caption** Photo caption (may also be used when resending photos by *file\_id*), 0-1024 characters after entities parsing
- parse\_mode Mode for parsing entities in the photo caption. See formatting options for more details.
- **caption\_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse\_mode*
- **show\_caption\_above\_media** Pass True, if the caption must be shown above the message media
- has\_spoiler Pass True if the photo needs to be covered with a spoiler animation
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.

- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found

instance of method aiogram.methods.send\_photo.SendPhoto

 $answer\_photo(photo: Union[InputFile, str], caption: Optional[str] = None, parse\_mode:$ 

 $Optional[Union[str, Default]] = \langle Default('parse_mode') \rangle, caption_entities: \\ Optional[List[MessageEntity]] = None, show_caption_above_media: Optional[Union[bool, Default]] = \langle Default('show_caption_above_media') \rangle, has_spoiler: Optional[bool] = None, \\ disable_notification: Optional[bool] = None, protect_content: Optional[Union[bool, Default]] = \langle Default('protect_content') \rangle, message_effect_id: Optional[str] = None, \\ reply_parameters: Optional[ReplyParameters] = None, reply_markup: \\ Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, \\ ForceReply]] = None, allow_sending_without_reply: Optional[bool] = None, \\ reply_to_message_id: Optional[int] = None, **kwargs: Any) \rightarrow SendPhoto$ 

Shortcut for method <code>aiogram.methods.send\_photo.SendPhoto</code> will automatically fill method attributes:

- · chat id
- message\_thread\_id
- business\_connection\_id

Use this method to send photos. On success, the sent aiogram.types.message.Message is returned.

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

## **Parameters**

- **photo** Photo to send. Pass a file\_id as String to send a photo that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a photo from the Internet, or upload a new photo using multipart/form-data. The photo must be at most 10 MB in size. The photo's width and height must not exceed 10000 in total. Width and height ratio must be at most 20. *More information on Sending Files* »
- **caption** Photo caption (may also be used when resending photos by *file\_id*), 0-1024 characters after entities parsing
- parse\_mode Mode for parsing entities in the photo caption. See formatting options for more details.
- **caption\_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse\_mode*
- **show\_caption\_above\_media** Pass True, if the caption must be shown above the message media
- has\_spoiler Pass True if the photo needs to be covered with a spoiler animation
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.

- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send\_photo.SendPhoto

```
reply_poll(question: str, options: List[Union[InputPollOption, str]], question_parse_mode:

Optional[Union[str, Default]] = <Default('parse_mode')>, question_entities:

Optional[List[MessageEntity]] = None, is_anonymous: Optional[bool] = None, type:

Optional[str] = None, allows_multiple_answers: Optional[bool] = None, correct_option_id:

Optional[int] = None, explanation: Optional[str] = None, explanation_parse_mode:

Optional[Union[str, Default]] = <Default('parse_mode')>, explanation_entities:

Optional[List[MessageEntity]] = None, open_period: Optional[int] = None, close_date:

Optional[Union[datetime.datetime, datetime.timedelta, int]] = None, is_closed: Optional[bool] = None, disable_notification: Optional[bool] = None, protect_content: Optional[Union[bool, Default]] = <Default('protect_content')>, message_effect_id: Optional[str] = None, reply_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None, allow_sending_without_reply: Optional[bool] = None, **kwargs: Any) → SendPoll
```

Shortcut for method aiogram.methods.send\_poll.SendPoll will automatically fill method attributes:

- chat\_id
- message\_thread\_id
- business\_connection\_id
- reply\_parameters

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

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

- **question** Poll question, 1-300 characters
- options A JSON-serialized list of 2-10 answer options
- **question\_parse\_mode** Mode for parsing entities in the question. See formatting options for more details. Currently, only custom emoji entities are allowed
- **question\_entities** A JSON-serialized list of special entities that appear in the poll question. It can be specified instead of *question\_parse\_mode*
- is\_anonymous True, if the poll needs to be anonymous, defaults to True
- type Poll type, 'quiz' or 'regular', defaults to 'regular'

- allows\_multiple\_answers True, if the poll allows multiple answers, ignored for polls in quiz mode, defaults to False
- **correct\_option\_id** 0-based identifier of the correct answer option, required for polls in quiz mode
- **explanation** 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** Mode for parsing entities in the explanation. See formatting options for more details.
- **explanation\_entities** A JSON-serialized list of special entities that appear in the poll explanation. It can be specified instead of *explanation\_parse\_mode*
- **open\_period** Amount of time in seconds the poll will be active after creation, 5-600. Can't be used together with *close\_date*.
- **close\_date** 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** Pass True if the poll needs to be immediately closed. This can be useful for poll preview.
- disable\_notification Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- **message\_effect\_id** Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found

instance of method aiogram.methods.send\_poll.SendPoll

```
answer_poll(question: str, options: List[Union[InputPollOption, str]], question_parse_mode:

Optional[Union[str, Default]] = <Default('parse_mode')>, question_entities:

Optional[List[MessageEntity]] = None, is_anonymous: Optional[bool] = None, type:

Optional[str] = None, allows_multiple_answers: Optional[bool] = None, correct_option_id:

Optional[int] = None, explanation: Optional[str] = None, explanation_parse_mode:

Optional[Union[str, Default]] = <Default('parse_mode')>, explanation_entities:

Optional[List[MessageEntity]] = None, open_period: Optional[int] = None, close_date:

Optional[Union[datetime.datetime, datetime.timedelta, int]] = None, is_closed:

Optional[bool] = None, disable_notification: Optional[bool] = None, protect_content:

Optional[Union[bool, Default]] = <Default('protect_content')>, message_effect_id:

Optional[str] = None, reply_parameters: Optional[ReplyParameters] = None, reply_markup:

Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove,
ForceReply]] = None, allow_sending_without_reply: Optional[bool] = None,
reply_to_message_id: Optional[int] = None, **kwargs: Any) → SendPoll
```

Shortcut for method aiogram.methods.send\_poll.SendPoll will automatically fill method attributes:

• chat\_id

- message\_thread\_id
- business\_connection\_id

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

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

- question Poll question, 1-300 characters
- options A JSON-serialized list of 2-10 answer options
- **question\_parse\_mode** Mode for parsing entities in the question. See formatting options for more details. Currently, only custom emoji entities are allowed
- **question\_entities** A JSON-serialized list of special entities that appear in the poll question. It can be specified instead of *question\_parse\_mode*
- is\_anonymous True, if the poll needs to be anonymous, defaults to True
- type Poll type, 'quiz' or 'regular', defaults to 'regular'
- **allows\_multiple\_answers** True, if the poll allows multiple answers, ignored for polls in quiz mode, defaults to False
- **correct\_option\_id** 0-based identifier of the correct answer option, required for polls in quiz mode
- **explanation** 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** Mode for parsing entities in the explanation. See formatting options for more details.
- **explanation\_entities** A JSON-serialized list of special entities that appear in the poll explanation. It can be specified instead of *explanation\_parse\_mode*
- **open\_period** Amount of time in seconds the poll will be active after creation, 5-600. Can't be used together with *close\_date*.
- **close\_date** 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** Pass True if the poll needs to be immediately closed. This can be useful for poll preview.
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_parameters** Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send\_poll.SendPoll

 $\begin{aligned} \textbf{reply\_dice}(\textit{emoji: Optional[str]} &= \textit{None, disable\_notification: Optional[bool]} &= \textit{None, protect\_content:} \\ &\quad \textit{Optional[Union[bool, Default]]} &= \textit{Optional['protect\_content')} >, \textit{message\_effect\_id:} \\ &\quad \textit{Optional[str]} &= \textit{None, reply\_markup: Optional[Union[InlineKeyboardMarkup,} \\ &\quad \textit{ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]}] &= \textit{None,} \\ &\quad \textit{allow\_sending\_without\_reply: Optional[bool]} &= \textit{None, **kwargs: Any}) \rightarrow \textit{SendDice} \end{aligned}$ 

Shortcut for method <code>aiogram.methods.send\_dice.SendDice</code> will automatically fill method attributes:

- chat\_id
- message\_thread\_id
- business\_connection\_id
- reply\_parameters

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

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

#### **Parameters**

- **emoji** 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** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found

### Returns

instance of method aiogram.methods.send\_dice.SendDice

 $\begin{tabular}{ll} \textbf{answer\_dice}(emoji: Optional[str] = None, disable\_notification: Optional[bool] = None, protect\_content: \\ Optional[Union[bool, Default]] = & Optional['] >, message\_effect\_id: \\ Optional[str] = None, reply\_parameters: Optional[ReplyParameters] = None, reply\_markup: \\ Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None, allow\_sending\_without\_reply: Optional[bool] = None, \\ reply\_to\_message\_id: Optional[int] = None, **kwargs: Any) \rightarrow SendDice \\ \end{tabular}$ 

Shortcut for method <code>aiogram.methods.send\_dice.SendDice</code> will automatically fill method attributes:

- chat\_id
- message\_thread\_id

business\_connection\_id

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

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

#### **Parameters**

- **emoji** 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** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding
- **message\_effect\_id** Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

# Returns

instance of method aiogram.methods.send\_dice.SendDice

```
reply_sticker(sticker: Union[InputFile, str], emoji: Optional[str] = None, disable_notification:

Optional[bool] = None, protect_content: Optional[Union[bool, Default]] =

<Default('protect_content')>, message_effect_id: Optional[str] = None, reply_markup:

Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove,
ForceReply]] = None, allow_sending_without_reply: Optional[bool] = None, **kwargs:

Any) → SendSticker
```

Shortcut for method <code>aiogram.methods.send\_sticker.SendSticker</code> will automatically fill method attributes:

- chat\_id
- message\_thread\_id
- business\_connection\_id
- reply\_parameters

Use this method to send static .WEBP, animated .TGS, or video .WEBM stickers. On success, the sent aiogram.types.message.Message is returned.

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

### **Parameters**

• **sticker** – Sticker to send. Pass a file\_id as String to send a file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a .WEBP sticker from the Internet, or upload a new .WEBP, .TGS, or .WEBM sticker using multipart/form-data. *More information on Sending Files* ». Video and animated stickers can't be sent via an HTTP URL.

- **emoji** Emoji associated with the sticker; only for just uploaded stickers
- disable\_notification Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found

instance of method aiogram.methods.send\_sticker.SendSticker

```
answer_sticker(sticker: Union[InputFile, str], emoji: Optional[str] = None, disable_notification:

Optional[bool] = None, protect_content: Optional[Union[bool, Default]] =

<Default('protect_content')>, message_effect_id: Optional[str] = None, reply_parameters:

Optional[ReplyParameters] = None, reply_markup:

Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove,
ForceReply]] = None, allow_sending_without_reply: Optional[bool] = None,
reply_to_message_id: Optional[int] = None, **kwargs: Any) → SendSticker
```

Shortcut for method <code>aiogram.methods.send\_sticker.SendSticker</code> will automatically fill method attributes:

- chat\_id
- message\_thread\_id
- business\_connection\_id

Use this method to send static .WEBP, animated .TGS, or video .WEBM stickers. On success, the sent aiogram.types.message.Message is returned.

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

## **Parameters**

- **sticker** Sticker to send. Pass a file\_id as String to send a file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a .WEBP sticker from the Internet, or upload a new .WEBP, .TGS, or .WEBM sticker using multipart/formdata. *More information on Sending Files* ». Video and animated stickers can't be sent via an HTTP URL.
- emoji Emoji associated with the sticker; only for just uploaded stickers
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- **message\_effect\_id** Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_parameters** Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send\_sticker.SendSticker

Shortcut for method aiogram.methods.send\_venue.SendVenue will automatically fill method attributes:

- chat\_id
- message\_thread\_id
- business\_connection\_id
- reply\_parameters

Use this method to send information about a venue. On success, the sent aiogram.types.message. Message is returned.

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

- latitude Latitude of the venue
- longitude Longitude of the venue
- title Name of the venue
- address Address of the venue
- **foursquare\_id** Foursquare identifier of the venue
- **foursquare\_type** Foursquare type of the venue, if known. (For example, 'arts\_entertainment/default', 'arts\_entertainment/aquarium' or 'food/icecream'.)
- google\_place\_id Google Places identifier of the venue
- **google\_place\_type** Google Places type of the venue. (See supported types.)
- disable\_notification Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found

instance of method aiogram.methods.send\_venue.SendVenue

answer\_venue(latitude: float, longitude: float, title: str, address: str, foursquare\_id: Optional[str] = None, foursquare\_type: Optional[str] = None, google\_place\_id: Optional[str] = None, google\_place\_type: Optional[str] = None, disable\_notification: Optional[bool] = None, protect\_content: Optional[Union[bool, Default]] = <Default('protect\_content')>, message\_effect\_id: Optional[str] = None, reply\_parameters: Optional[ReplyParameters] = None, reply\_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None, allow\_sending\_without\_reply: Optional[bool] = None, reply\_to\_message\_id: Optional[int] = None, \*\*kwargs: Any) → SendVenue

Shortcut for method aiogram.methods.send\_venue.SendVenue will automatically fill method attributes:

- chat\_id
- message\_thread\_id
- business\_connection\_id

Use this method to send information about a venue. On success, the sent aiogram.types.message. Message is returned.

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

## **Parameters**

- latitude Latitude of the venue
- longitude Longitude of the venue
- title Name of the venue
- address Address of the venue
- **foursquare\_id** Foursquare identifier of the venue
- **foursquare\_type** Foursquare type of the venue, if known. (For example, 'arts\_entertainment/default', 'arts\_entertainment/aquarium' or 'food/icecream'.)
- google\_place\_id Google Places identifier of the venue
- **google\_place\_type** Google Places type of the venue. (See supported types.)
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_parameters** Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

# Returns

instance of method aiogram.methods.send\_venue.SendVenue

```
reply_video(video: Union[InputFile, str], duration: Optional[int] = None, width: Optional[int] = None, height: Optional[int] = None, thumbnail: Optional[InputFile] = None, caption: Optional[str] = None, parse_mode: Optional[Union[str, Default]] = <Default('parse_mode')>, caption_entities: Optional[List[MessageEntity]] = None, show_caption_above_media: Optional[Union[bool, Default]] = <Default('show_caption_above_media')>, has_spoiler: Optional[bool] = None, supports_streaming: Optional[bool] = None, disable_notification: Optional[bool] = None, protect_content: Optional[Union[bool, Default]] = <Default('protect_content')>, message_effect_id: Optional[str] = None, reply_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None, allow_sending_without_reply: Optional[bool] = None, **kwargs: Any) → SendVideo
```

Shortcut for method <code>aiogram.methods.send\_video.SendVideo</code> will automatically fill method attributes:

- chat id
- message\_thread\_id
- business\_connection\_id
- reply\_parameters

Use this method to send video files, Telegram clients support MPEG4 videos (other formats may be sent as aiogram.types.document.Document). On success, the sent aiogram.types.message.Message is returned. Bots can currently send video files of up to 50 MB in size, this limit may be changed in the future.

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

- **video** Video to send. Pass a file\_id as String to send a video that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a video from the Internet, or upload a new video using multipart/form-data. *More information on Sending Files* »
- duration Duration of sent video in seconds
- width Video width
- height Video height
- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file\_attach\_name>' if the thumbnail was uploaded using multipart/form-data under <file\_attach\_name>. *More information on Sending Files*"""
- **caption** Video caption (may also be used when resending videos by *file\_id*), 0-1024 characters after entities parsing
- parse\_mode Mode for parsing entities in the video caption. See formatting options for more details.
- caption\_entities A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse\_mode*
- **show\_caption\_above\_media** Pass True, if the caption must be shown above the message media

- has\_spoiler Pass True if the video needs to be covered with a spoiler animation
- supports\_streaming Pass True if the uploaded video is suitable for streaming
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- **message\_effect\_id** Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found

instance of method aiogram.methods.send\_video.SendVideo

```
answer_video(video: Union[InputFile, str], duration: Optional[int] = None, width: Optional[int] = None, height: Optional[int] = None, thumbnail: Optional[InputFile] = None, caption:

Optional[str] = None, parse_mode: Optional[Union[str, Default]] =

<Default('parse_mode')>, caption_entities: Optional[List[MessageEntity]] = None,
show_caption_above_media: Optional[Union[bool, Default]] =

<Default('show_caption_above_media')>, has_spoiler: Optional[bool] = None,
supports_streaming: Optional[bool] = None, disable_notification: Optional[bool] = None,
protect_content: Optional[Union[bool, Default]] = <Default('protect_content')>,
message_effect_id: Optional[str] = None, reply_parameters: Optional[ReplyParameters] =
None, reply_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup,
ReplyKeyboardRemove, ForceReply]] = None, allow_sending_without_reply: Optional[bool]
= None, reply_to_message_id: Optional[int] = None, **kwargs: Any) → SendVideo
```

Shortcut for method aiogram.methods.send\_video.SendVideo will automatically fill method attributes:

- chat\_id
- message\_thread\_id
- business\_connection\_id

Use this method to send video files, Telegram clients support MPEG4 videos (other formats may be sent as <code>aiogram.types.document.Document</code>). On success, the sent <code>aiogram.types.message.Message</code> is returned. Bots can currently send video files of up to 50 MB in size, this limit may be changed in the future.

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

## **Parameters**

- **video** Video to send. Pass a file\_id as String to send a video that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a video from the Internet, or upload a new video using multipart/form-data. *More information on Sending Files* »
- duration Duration of sent video in seconds
- width Video width
- **height** Video height

- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file\_attach\_name>' if the thumbnail was uploaded using multipart/form-data under <file\_attach\_name>. *More information on Sending Files*»
- **caption** Video caption (may also be used when resending videos by *file\_id*), 0-1024 characters after entities parsing
- parse\_mode Mode for parsing entities in the video caption. See formatting options for more details.
- **caption\_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse\_mode*
- **show\_caption\_above\_media** Pass True, if the caption must be shown above the message media
- has\_spoiler Pass True if the video needs to be covered with a spoiler animation
- **supports\_streaming** Pass True if the uploaded video is suitable for streaming
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send\_video.SendVideo

**reply\_video\_note**(video\_note: Union[InputFile, str], duration: Optional[int] = None, length:

Optional[int] = None, thumbnail: Optional[InputFile] = None, disable notification:

Optional[bool] = None, protect content: Optional[Union[bool, Default]] =

<Default('protect\_content')>, message\_effect\_id: Optional[str] = None, reply\_markup:

Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup,

*ReplyKeyboardRemove, ForceReply*]] = *None, allow\_sending\_without\_reply:* 

 $Optional[bool] = None, **kwargs: Any) \rightarrow SendVideoNote$ 

Shortcut for method  $aiogram.methods.send\_video\_note.SendVideoNote$  will automatically fill method attributes:

- chat\_id
- message\_thread\_id
- business\_connection\_id
- reply\_parameters

As of v.4.0, Telegram clients support rounded square MPEG4 videos of up to 1 minute long. Use this method to send video messages. On success, the sent aiogram.types.message.Message is returned.

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

#### **Parameters**

- video\_note Video note to send. Pass a file\_id as String to send a video note that exists
  on the Telegram servers (recommended) or upload a new video using multipart/form-data.
   *More information on Sending Files* ». Sending video notes by a URL is currently unsupported
- duration Duration of sent video in seconds
- length Video width and height, i.e. diameter of the video message
- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file\_attach\_name>' if the thumbnail was uploaded using multipart/form-data under <file\_attach\_name>. *More information on Sending Files*
- disable\_notification Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found

# Returns

instance of method aiogram.methods.send\_video\_note.SendVideoNote

```
 \begin{tabular}{ll} \textbf{answer\_video\_note} (video\_note: Union[InputFile, str], duration: Optional[int] = None, length: \\ Optional[int] = None, thumbnail: Optional[InputFile] = None, disable\_notification: \\ Optional[bool] = None, protect\_content: Optional[Union[bool, Default]] = \\ < Default('protect\_content') >, message\_effect\_id: Optional[str] = None, \\ reply\_parameters: Optional[ReplyParameters] = None, reply\_markup: \\ Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, \\ ReplyKeyboardRemove, ForceReply]] = None, allow\_sending\_without\_reply: \\ Optional[bool] = None, reply\_to\_message\_id: Optional[int] = None, **kwargs: Any) \\ \rightarrow SendVideoNote \end{tabular}
```

Shortcut for method <code>aiogram.methods.send\_video\_note.SendVideoNote</code> will automatically fill method attributes:

- chat\_id
- message\_thread\_id
- business\_connection\_id

As of v.4.0, Telegram clients support rounded square MPEG4 videos of up to 1 minute long. Use this method to send video messages. On success, the sent aiogram.types.message.Message is returned.

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

#### **Parameters**

- video\_note Video note to send. Pass a file\_id as String to send a video note that exists on the Telegram servers (recommended) or upload a new video using multipart/form-data. *More information on Sending Files* ». Sending video notes by a URL is currently unsupported
- **duration** Duration of sent video in seconds
- length Video width and height, i.e. diameter of the video message
- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file\_attach\_name>' if the thumbnail was uploaded using multipart/form-data under <file\_attach\_name>. *More information on Sending Files*
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

# Returns

instance of method aiogram.methods.send\_video\_note.SendVideoNote

```
reply_voice(voice: Union[InputFile, str], caption: Optional[str] = None, parse_mode: Optional[Union[str, Default]] = <Default('parse_mode')>, caption_entities: Optional[List[MessageEntity]] = None, duration: Optional[int] = None, disable_notification: Optional[bool] = None, protect_content: Optional[Union[bool, Default]] = <Default('protect_content')>, message_effect_id: Optional[str] = None, reply_markup:
Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None, allow_sending_without_reply: Optional[bool] = None, **kwargs: Any) → SendVoice
```

Shortcut for method aiogram.methods.send\_voice.SendVoice will automatically fill method attributes:

- chat\_id
- message\_thread\_id
- business\_connection\_id
- reply\_parameters

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, or in .MP3 format, or in .M4A format (other formats may be sent as <code>aiogram.types.audio.Audio</code> or <code>aiogram.types.document.Document</code>). On success, the sent <code>aiogram.types.message</code> is returned. Bots can currently send voice messages of up to 50 MB in size, this limit may be changed in the future.

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

#### **Parameters**

- **voice** Audio file to send. Pass a file\_id as String to send a file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a file from the Internet, or upload a new one using multipart/form-data. *More information on Sending Files* »
- caption Voice message caption, 0-1024 characters after entities parsing
- parse\_mode Mode for parsing entities in the voice message caption. See formatting options for more details.
- caption\_entities A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse\_mode*
- duration Duration of the voice message in seconds
- disable\_notification Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found

## Returns

instance of method aiogram.methods.send\_voice.SendVoice

```
answer_voice(voice: Union[InputFile, str], caption: Optional[str] = None, parse_mode:

Optional[Union[str, Default]] = <Default('parse_mode')>, caption_entities:

Optional[List[MessageEntity]] = None, duration: Optional[int] = None, disable_notification:

Optional[bool] = None, protect_content: Optional[Union[bool, Default]] =

<Default('protect_content')>, message_effect_id: Optional[str] = None, reply_parameters:

Optional[ReplyParameters] = None, reply_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None,

allow_sending_without_reply: Optional[bool] = None, reply_to_message_id: Optional[int] =

None, **kwargs: Any) → SendVoice
```

Shortcut for method <code>aiogram.methods.send\_voice.SendVoice</code> will automatically fill method attributes:

- chat id
- message\_thread\_id
- business\_connection\_id

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, or in .MP3 format,

or in .M4A format (other formats may be sent as aiogram.types.audio.Audio or aiogram.types. document. Document). On success, the sent aiogram. types.message. Message is returned. Bots can currently send voice messages of up to 50 MB in size, this limit may be changed in the future.

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

## **Parameters**

- voice Audio file to send. Pass a file\_id as String to send a file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a file from the Internet, or upload a new one using multipart/form-data. More information on Sending Files »
- caption Voice message caption, 0-1024 characters after entities parsing
- parse\_mode Mode for parsing entities in the voice message caption. See formatting options for more details.
- caption\_entities A JSON-serialized list of special entities that appear in the caption, which can be specified instead of parse\_mode
- **duration** Duration of the voice message in seconds
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- message\_effect\_id Unique identifier of the message effect to be added to the message; for private chats only
- reply\_parameters Description of the message to reply to
- reply\_markup Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

# **Returns**

instance of method aiogram.methods.send\_voice.SendVoice

**send\_copy** (chat\_id: str | int, disable\_notification: bool | None = None, reply\_to\_message\_id: int | None = None, reply\_parameters: ReplyParameters | None = None, reply\_markup: InlineKeyboardMarkup | ReplyKeyboardMarkup | None = None, allow\_sending\_without\_reply: bool | None = None, message thread id: int | None = None, business connection id: str | None = None, parse mode:  $str \mid None = None, message \ effect \ id: str \mid None = None) \rightarrow ForwardMessage \mid SendAnimation \mid$ SendAudio | SendContact | SendDocument | SendLocation | SendMessage | SendPhoto | SendPoll | SendDice | SendSticker | SendVenue | SendVideo | SendVideoNote | SendVoice

Send copy of a message.

Is similar to aiogram.client.bot.Bot.copy\_message() but returning the sent message instead of aiogram.types.message\_id.MessageId



# 1 Note

This method doesn't use the API method named *copyMessage* and historically implemented before the similar method is added to API

#### **Parameters**

- · chat id
- disable\_notification
- reply\_to\_message\_id
- reply\_parameters
- reply\_markup
- allow\_sending\_without\_reply
- message\_thread\_id
- business\_connection\_id
- parse\_mode
- message\_effect\_id

#### **Returns**

copy\_to(chat\_id: Union[int, str], message\_thread\_id: Optional[int] = None, caption: Optional[str] = None, parse\_mode: Optional[Union[str, Default]] = <Default('parse\_mode')>, caption\_entities: Optional[List[MessageEntity]] = None, show\_caption\_above\_media: Optional[Union[bool, Default]] = <Default('show\_caption\_above\_media')>, disable\_notification: Optional[bool] = None, protect\_content: Optional[Union[bool, Default]] = <Default('protect\_content')>, reply\_parameters: Optional[ReplyParameters] = None, reply\_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None, allow\_sending\_without\_reply: Optional[bool] = None, reply\_to\_message\_id: Optional[int] = None, \*\*kwargs: Any) → CopyMessage

Shortcut for method  $aiogram.methods.copy\_message.CopyMessage$  will automatically fill method attributes:

- from\_chat\_id
- message\_id

Use this method to copy messages of any kind. Service messages, paid media messages, giveaway messages, giveaway winners messages, and invoice messages can't be copied. A quiz aiogram.methods. poll.Poll can be copied only if the value of the field <code>correct\_option\_id</code> is known to the bot. The method is analogous to the method <code>aiogram.methods.forward\_message.ForwardMessage</code>, but the copied message doesn't have a link to the original message. Returns the <code>aiogram.types.message\_id.MessageId</code> of the sent message on success.

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

#### **Parameters**

- **chat\_id** Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- **message\_thread\_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **caption** New caption for media, 0-1024 characters after entities parsing. If not specified, the original caption is kept
- parse\_mode Mode for parsing entities in the new caption. See formatting options for more details.

- **caption\_entities** A JSON-serialized list of special entities that appear in the new caption, which can be specified instead of *parse mode*
- **show\_caption\_above\_media** Pass True, if the caption must be shown above the message media. Ignored if a new caption isn't specified.
- disable\_notification Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow\_sending\_without\_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply\_to\_message\_id If the message is a reply, ID of the original message

instance of method aiogram.methods.copy\_message.CopyMessage

edit\_text (text: str, business\_connection\_id: Optional[str] = None, inline\_message\_id: Optional[str] = None, parse\_mode: Optional[Union[str, Default]] = <Default('parse\_mode')>, entities: Optional[List[MessageEntity]] = None, link\_preview\_options: Optional[Union[LinkPreviewOptions, Default]] = <Default('link\_preview')>, reply\_markup: Optional[InlineKeyboardMarkup] = None, disable\_web\_page\_preview: Optional[Union[bool, Default]] = <Default('link\_preview\_is\_disabled')>, \*\*kwargs: Any) → EditMessageText

Shortcut for method <code>aiogram.methods.edit\_message\_text.EditMessageText</code> will automatically fill method attributes:

- chat\_id
- message\_id

Use this method to edit text and game messages. On success, if the edited message is not an inline message, the edited <code>aiogram.types.message</code> is returned, otherwise <code>True</code> is returned. Note that business messages that were not sent by the bot and do not contain an inline keyboard can only be edited within <code>48</code> hours from the time they were sent.

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

### **Parameters**

- **text** New text of the message, 1-4096 characters after entities parsing
- business\_connection\_id Unique identifier of the business connection on behalf of which the message to be edited was sent
- inline\_message\_id Required if chat\_id and message\_id are not specified. Identifier
  of the inline message
- parse\_mode Mode for parsing entities in the message text. See formatting options for more details.
- **entities** A JSON-serialized list of special entities that appear in message text, which can be specified instead of *parse\_mode*
- link\_preview\_options Link preview generation options for the message
- reply\_markup A JSON-serialized object for an inline keyboard.

• disable\_web\_page\_preview – Disables link previews for links in this message

#### Returns

instance of method aiogram.methods.edit\_message\_text.EditMessageText

**forward**(chat\_id: Union[int, str], message\_thread\_id: Optional[int] = None, disable\_notification: Optional[bool] = None, protect\_content: Optional[Union[bool, Default]] = <Default('protect\_content')>, \*\*kwargs: Any) → ForwardMessage

Shortcut for method aiogram.methods.forward\_message.ForwardMessage will automatically fill method attributes:

- from\_chat\_id
- message\_id

Use this method to forward messages of any kind. Service messages and messages with protected content can't be forwarded. On success, the sent aiogram.types.message.Message is returned.

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

#### **Parameters**

- chat\_id Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- message\_thread\_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- **protect\_content** Protects the contents of the forwarded message from forwarding and saving

### Returns

instance of method aiogram.methods.forward\_message.ForwardMessage

edit\_media(media: InputMediaAnimation | InputMediaDocument | InputMediaAudio | InputMediaPhoto | InputMediaVideo, business\_connection\_id: str | None = None, inline\_message\_id: str | None = None, reply\_markup: InlineKeyboardMarkup | None = None, \*\*kwargs: Any) → EditMessageMedia

Shortcut for method <code>aiogram.methods.edit\_message\_media.EditMessageMedia</code> will automatically fill method attributes:

- chat\_id
- message\_id

Use this method to edit animation, audio, document, photo, or video messages. If a message is part of a message album, then it can be edited only to an audio for audio albums, only to a document for document albums and to a photo or a video otherwise. When an inline message is edited, a new file can't be uploaded; use a previously uploaded file via its file\_id or specify a URL. On success, if the edited message is not an inline message, the edited <code>aiogram.types.message.Message</code> is returned, otherwise True is returned. Note that business messages that were not sent by the bot and do not contain an inline keyboard can only be edited within 48 hours from the time they were sent.

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

## **Parameters**

• media – A JSON-serialized object for a new media content of the message

- business\_connection\_id Unique identifier of the business connection on behalf of which the message to be edited was sent
- **inline\_message\_id** Required if *chat\_id* and *message\_id* are not specified. Identifier of the inline message
- reply\_markup A JSON-serialized object for a new inline keyboard.

instance of method aiogram.methods.edit\_message\_media.EditMessageMedia

```
edit_reply_markup(business_connection_id: str | None = None, inline_message_id: str | None = None, reply_markup: InlineKeyboardMarkup | None = None, **kwargs: Any) → EditMessageReplyMarkup
```

Shortcut for method aiogram.methods.edit\_message\_reply\_markup.EditMessageReplyMarkup will automatically fill method attributes:

- chat\_id
- message\_id

Use this method to edit only the reply markup of messages. On success, if the edited message is not an inline message, the edited <code>aiogram.types.message.Message</code> is returned, otherwise True is returned. Note that business messages that were not sent by the bot and do not contain an inline keyboard can only be edited within **48 hours** from the time they were sent.

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

#### **Parameters**

- business\_connection\_id Unique identifier of the business connection on behalf of which the message to be edited was sent
- **inline\_message\_id** Required if *chat\_id* and *message\_id* are not specified. Identifier of the inline message
- reply\_markup A JSON-serialized object for an inline keyboard.

#### Returns

```
instance of method aiogram.methods.edit_message_reply_markup.
EditMessageReplyMarkup
```

```
delete_reply_markup(business_connection_id: str \mid None = None, inline\_message\_id: str \mid None = None, **kwargs: Any) \rightarrow EditMessageReplyMarkup
```

Shortcut for method  $aiogram.methods.edit\_message\_reply\_markup.EditMessageReplyMarkup$  will automatically fill method attributes:

- chat\_id
- message\_id
- reply\_markup

Use this method to edit only the reply markup of messages. On success, if the edited message is not an inline message, the edited <code>aiogram.types.message.Message</code> is returned, otherwise True is returned. Note that business messages that were not sent by the bot and do not contain an inline keyboard can only be edited within 48 hours from the time they were sent.

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

#### **Parameters**

• business\_connection\_id — Unique identifier of the business connection on behalf of which the message to be edited was sent

inline\_message\_id – Required if chat\_id and message\_id are not specified. Identifier
of the inline message

#### Returns

instance of method aiogram.methods.edit\_message\_reply\_markup.
EditMessageReplyMarkup

edit\_live\_location(latitude: float, longitude: float, business\_connection\_id: str | None = None, inline\_message\_id: str | None = None, live\_period: int | None = None, horizontal\_accuracy: float | None = None, heading: int | None = None, proximity\_alert\_radius: int | None = None, reply\_markup: InlineKeyboardMarkup | None = None, \*\*kwargs: Any) → EditMessageLiveLocation

Shortcut for method aiogram.methods.edit\_message\_live\_location. EditMessageLiveLocation will automatically fill method attributes:

- chat\_id
- message\_id

Use this method to edit live location messages. A location can be edited until its <code>live\_period</code> expires or editing is explicitly disabled by a call to <code>aiogram.methods.stop\_message\_live\_location</code>. StopMessageLiveLocation. On success, if the edited message is not an inline message, the edited <code>aiogram.types.message.Message</code> is returned, otherwise <code>True</code> is returned.

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

#### **Parameters**

- latitude Latitude of new location
- longitude Longitude of new location
- business\_connection\_id Unique identifier of the business connection on behalf of which the message to be edited was sent
- inline\_message\_id Required if chat\_id and message\_id are not specified. Identifier
  of the inline message
- **live\_period** New period in seconds during which the location can be updated, starting from the message send date. If 0x7FFFFFFF is specified, then the location can be updated forever. Otherwise, the new value must not exceed the current *live\_period* by more than a day, and the live location expiration date must remain within the next 90 days. If not specified, then *live\_period* remains unchanged
- **horizontal\_accuracy** The radius of uncertainty for the location, measured in meters; 0-1500
- **heading** Direction in which the user is moving, in degrees. Must be between 1 and 360 if specified.
- **proximity\_alert\_radius** The maximum distance for proximity alerts about approaching another chat member, in meters. Must be between 1 and 100000 if specified.
- **reply\_markup** A JSON-serialized object for a new inline keyboard.

### Returns

 $\begin{array}{lll} \text{instance} & \text{of} & \text{method} & \textit{aiogram.methods.edit\_message\_live\_location}. \\ \textit{EditMessageLiveLocation} & \end{array}$ 

Shortcut for method aiogram.methods.stop\_message\_live\_location. StopMessageLiveLocation will automatically fill method attributes:

- chat\_id
- message\_id

Use this method to stop updating a live location message before *live\_period* expires. On success, if the message is not an inline message, the edited *aiogram.types.message.Message* is returned, otherwise True is returned.

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

#### **Parameters**

- business\_connection\_id Unique identifier of the business connection on behalf of which the message to be edited was sent
- **inline\_message\_id** Required if *chat\_id* and *message\_id* are not specified. Identifier of the inline message
- **reply\_markup** A JSON-serialized object for a new inline keyboard.

#### Returns

instance of method aiogram.methods.stop\_message\_live\_location.
StopMessageLiveLocation

Shortcut for method aiogram.methods.edit\_message\_caption.EditMessageCaption will automatically fill method attributes:

- chat\_id
- message\_id

Use this method to edit captions of messages. On success, if the edited message is not an inline message, the edited <code>aiogram.types.message</code> is returned, otherwise True is returned. Note that business messages that were not sent by the bot and do not contain an inline keyboard can only be edited within 48 hours from the time they were sent.

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

#### **Parameters**

- business\_connection\_id Unique identifier of the business connection on behalf of which the message to be edited was sent
- **inline\_message\_id** Required if *chat\_id* and *message\_id* are not specified. Identifier of the inline message
- caption New caption of the message, 0-1024 characters after entities parsing
- parse\_mode Mode for parsing entities in the message caption. See formatting options for more details.
- **caption\_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse\_mode*

- **show\_caption\_above\_media** Pass True, if the caption must be shown above the message media. Supported only for animation, photo and video messages.
- **reply\_markup** A JSON-serialized object for an inline keyboard.

instance of method aiogram.methods.edit\_message\_caption.EditMessageCaption

 $delete(**kwargs: Any) \rightarrow DeleteMessage$ 

Shortcut for method <code>aiogram.methods.delete\_message.DeleteMessage</code> will automatically fill method attributes:

- chat\_id
- message\_id

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.
- Service messages about a supergroup, channel, or forum topic creation can't be deleted.
- A dice message in a private chat can only be deleted if it was sent more than 24 hours ago.
- Bots can delete outgoing messages in private chats, groups, and supergroups.
- Bots can delete incoming messages in private chats.
- Bots granted can\_post\_messages permissions can delete outgoing messages in channels.
- If the bot is an administrator of a group, it can delete any message there.
- If the bot has can\_delete\_messages permission in a supergroup or a channel, it can delete any message there.

Returns True on success.

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

#### Returns

instance of method aiogram.methods.delete\_message.DeleteMessage

 $pin(business\_connection\_id: str \mid None = None, disable\_notification: bool \mid None = None, **kwargs: Any) \rightarrow PinChatMessage$ 

Shortcut for method <code>aiogram.methods.pin\_chat\_message.PinChatMessage</code> will automatically fill method attributes:

- chat\_id
- message\_id

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' administrator right in a supergroup or 'can\_edit\_messages' administrator right in a channel. Returns True on success.

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

#### **Parameters**

- business\_connection\_id Unique identifier of the business connection on behalf of which the message will be pinned
- **disable\_notification** Pass True if it is not necessary to send a notification to all chat members about the new pinned message. Notifications are always disabled in channels and private chats.

instance of method aiogram.methods.pin\_chat\_message.PinChatMessage

**unpin**(business\_connection\_id:  $str \mid None = None, **kwargs: Any) \rightarrow UnpinChatMessage$ 

Shortcut for method aiogram.methods.unpin\_chat\_message.UnpinChatMessage will automatically fill method attributes:

- · chat id
- message\_id

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' administrator right in a supergroup or 'can\_edit\_messages' administrator right in a channel. Returns True on success.

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

#### **Parameters**

**business\_connection\_id** – Unique identifier of the business connection on behalf of which the message will be unpinned

#### Returns

instance of method aiogram.methods.unpin\_chat\_message.UnpinChatMessage

 $get\_url(force\_private: bool = False, include\_thread\_id: bool = False) \rightarrow str | None$ 

Returns message URL. Cannot be used in private (one-to-one) chats. If chat has a username, returns URL like https://t.me/username/message\_id Otherwise (or if {force\_private} flag is set), returns https://t.me/c/shifted chat id/message id

#### **Parameters**

- force\_private if set, a private URL is returned even for a public chat
- include\_thread\_id if set, adds chat thread id to URL and returns like https://t.me/username/thread\_id/message\_id

# Returns

string with full message URL

**react**(*reaction: List*[ReactionTypeEmoji | ReactionTypeCustomEmoji | ReactionTypePaid] | *None* = *None*, *is\_big: bool* | *None* = *None*, \*\*kwargs: Any) → *SetMessageReaction* 

Shortcut for method aiogram.methods.set\_message\_reaction.SetMessageReaction will automatically fill method attributes:

- chat\_id
- message\_id

Use this method to change the chosen reactions on a message. Service messages can't be reacted to. Automatically forwarded messages from a channel to its discussion group have the same available reactions as messages in the channel. Bots can't use paid reactions. Returns True on success.

Source: https://core.telegram.org/bots/api#setmessagereaction

### **Parameters**

- **reaction** A JSON-serialized list of reaction types to set on the message. Currently, as non-premium users, bots can set up to one reaction per message. A custom emoji reaction can be used if it is either already present on the message or explicitly allowed by chat administrators. Paid reactions can't be used by bots.
- **is\_big** Pass True to set the reaction with a big animation

instance of method aiogram.methods.set\_message\_reaction.SetMessageReaction

answer\_paid\_media(star\_count: int, media: List[InputPaidMediaPhoto | InputPaidMediaVideo], payload:

str | None = None, caption: str | None = None, parse\_mode: str | None = None,

caption\_entities: List[MessageEntity] | None = None, show\_caption\_above\_media:

bool | None = None, disable\_notification: bool | None = None, protect\_content: bool |

None = None, reply\_parameters: ReplyParameters | None = None, reply\_markup:

InlineKeyboardMarkup | ReplyKeyboardMarkup | ReplyKeyboardRemove |

ForceReply | None = None, \*\*kwargs: Any) → SendPaidMedia

Shortcut for method aiogram.methods.send\_paid\_media.SendPaidMedia will automatically fill method attributes:

- chat\_id
- message\_thread\_id
- business\_connection\_id

Use this method to send paid media. On success, the sent aiogram.types.message.Message is returned.

Source: https://core.telegram.org/bots/api#sendpaidmedia

#### **Parameters**

- **star\_count** The number of Telegram Stars that must be paid to buy access to the media; 1-2500
- media A JSON-serialized array describing the media to be sent; up to 10 items
- **payload** Bot-defined paid media payload, 0-128 bytes. This will not be displayed to the user, use it for your internal processes.
- caption Media caption, 0-1024 characters after entities parsing
- parse\_mode Mode for parsing entities in the media caption. See formatting options for more details.
- **caption\_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse\_mode*
- **show\_caption\_above\_media** Pass True, if the caption must be shown above the message media
- disable\_notification Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- reply\_parameters Description of the message to reply to
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

### Returns

instance of method aiogram.methods.send\_paid\_media.SendPaidMedia

Shortcut for method aiogram.methods.send\_paid\_media.SendPaidMedia will automatically fill method attributes:

- chat\_id
- message\_thread\_id
- business\_connection\_id
- reply\_parameters

Use this method to send paid media. On success, the sent aiogram. types.message.Message is returned.

Source: https://core.telegram.org/bots/api#sendpaidmedia

#### **Parameters**

- **star\_count** The number of Telegram Stars that must be paid to buy access to the media; 1-2500
- media A JSON-serialized array describing the media to be sent; up to 10 items
- **payload** Bot-defined paid media payload, 0-128 bytes. This will not be displayed to the user, use it for your internal processes.
- caption Media caption, 0-1024 characters after entities parsing
- parse\_mode Mode for parsing entities in the media caption. See formatting options for more details.
- **caption\_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse\_mode*
- **show\_caption\_above\_media** Pass True, if the caption must be shown above the message media
- **disable\_notification** Sends the message silently. Users will receive a notification with no sound.
- protect\_content Protects the contents of the sent message from forwarding and saving
- **reply\_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

### Returns

instance of method aiogram.methods.send\_paid\_media.SendPaidMedia

## MessageAutoDeleteTimerChanged

```
class aiogram.types.message_auto_delete_timer_changed.MessageAutoDeleteTimerChanged(*,
```

message\_auto\_delete\_time:
int,
\*\*extra\_data:
Any)

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

Source: https://core.telegram.org/bots/api#messageautodeletetimerchanged

```
message_auto_delete_time: int
```

New auto-delete time for messages in the chat; in seconds

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

## MessageEntity

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

Source: https://core.telegram.org/bots/api#messageentity

```
type: str
```

Type of the entity. Currently, can be 'mention' (@username), 'hashtag' (#hashtag), 'cashtag' (\$USD), 'bot\_command' (/start@jobs\_bot), 'url' (https://telegram.org), 'email' (do-not-reply@telegram.org), 'phone\_number' (+1-212-555-0123), 'bold' (bold text), 'italic' (italic text), 'underline' (underlined text), 'strikethrough' (strikethrough text), 'spoiler' (spoiler message), 'blockquote' (block quotation), 'expandable\_blockquote' (collapsed-by-default block quotation), 'code' (monowidth string), 'pre' (monowidth block), 'text\_link' (for clickable text URLs), 'text\_mention' (for users without usernames), 'custom\_emoji' (for inline custom emoji stickers)

### offset: int

Offset in UTF-16 code units to the start of the entity

### length: int

Length of the entity in UTF-16 code units

#### url: str | None

Optional. For 'text\_link' only, URL that will be opened after user taps on the text

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
user: User | None
      Optional. For 'text_mention' only, the mentioned user

language: str | None
      Optional. For 'pre' only, the programming language of the entity text

custom_emoji_id: str | None
      Optional. For 'custom_emoji' only, unique identifier of the custom emoji. Use aiogram.methods.
      get_custom_emoji_stickers.GetCustomEmojiStickers to get full information about the sticker

extract_from(text: str) → str
```

## Messageld

```
class aiogram.types.message_id.MessageId(*, message_id: int, **extra_data: Any)
```

This object represents a unique message identifier.

Source: https://core.telegram.org/bots/api#messageid

```
message_id: int
```

Unique message identifier

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

### MessageOrigin

```
class aiogram.types.message_origin.MessageOrigin(**extra_data: Any)
```

This object describes the origin of a message. It can be one of

- aiogram.types.message\_origin\_user.MessageOriginUser
- aiogram.types.message\_origin\_hidden\_user.MessageOriginHiddenUser
- aiogram.types.message\_origin\_chat.MessageOriginChat
- aiogram.types.message\_origin\_channel.MessageOriginChannel

Source: https://core.telegram.org/bots/api#messageorigin

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

## MessageOriginChannel

```
class aiogram.types.message_origin_channel.MessageOriginChannel(*, type: Lit-
eral[MessageOriginType.CHANNEL]
=
MessageOriginType.CHANNEL,
date: _datetime_serializer,
return_type=int,
when_used=unless - none)],
chat: Chat, message_id: int,
author_signature: str | None =
None, **extra_data: Any)

The message was originally sent to a channel chat.
Source: https://core.telegram.org/bots/api#messageoriginchannel
```

type: Literal[MessageOriginType.CHANNEL]

Type of the message origin, always 'channel'

date: DateTime

Date the message was sent originally in Unix time

chat: Chat

Channel chat to which the message was originally sent

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
message_id: int
```

Unique message identifier inside the chat

```
author_signature: str | None
```

Optional. Signature of the original post author

## MessageOriginChat

```
class aiogram.types.message_origin_chat.MessageOriginChat(*, type:
```

```
Literal[MessageOriginType.CHAT] =
MessageOriginType.CHAT, date:
_datetime_serializer, return_type=int,
when_used=unless - none)],
sender_chat: Chat, author_signature: str
| None = None, **extra_data: Any)
```

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

Source: https://core.telegram.org/bots/api#messageoriginchat

## type: Literal[MessageOriginType.CHAT]

Type of the message origin, always 'chat'

date: DateTime

Date the message was sent originally in Unix time

### model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### sender\_chat: Chat

Chat that sent the message originally

```
author_signature: str | None
```

Optional. For messages originally sent by an anonymous chat administrator, original message author signature

# MessageOriginHiddenUser

```
class aiogram.types.message_origin_hidden_user.MessageOriginHiddenUser(*, type: Lit-
```

eral[MessageOriginType.HIDDEN\_USER
= MessageOriginType.HIDDEN\_USER,
date:
\_datetime\_serializer,
return\_type=int,
when\_used=unless none)],
sender\_user\_name: str,
\*\*extra\_data: Any)

The message was originally sent by an unknown user.

Source: https://core.telegram.org/bots/api#messageoriginhiddenuser

### type: Literal[MessageOriginType.HIDDEN\_USER]

Type of the message origin, always 'hidden\_user'

### date: \_datetime\_serializer, return\_type=int, when\_used=unless-none)]

Date the message was sent originally in Unix time

### model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model post init method.

#### sender\_user\_name: str

Name of the user that sent the message originally

### MessageOriginUser

```
class aiogram.types.message_origin_user.MessageOriginUser(*, type:
```

```
Literal[MessageOriginType.USER] =
MessageOriginType.USER, date:
_datetime_serializer, return_type=int,
when_used=unless - none)],
sender_user: User, **extra_data: Any)
```

The message was originally sent by a known user.

Source: https://core.telegram.org/bots/api#messageoriginuser

## type: Literal[MessageOriginType.USER]

Type of the message origin, always 'user'

date: DateTime

Date the message was sent originally in Unix time

### model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

sender\_user: *User* 

User that sent the message originally

## MessageReactionCountUpdated

```
class aiogram.types.message_reaction_count_updated.MessageReactionCountUpdated(*, chat: Chat,
```

```
message_id:
int, date:
__date-
time_serializer,
re-
turn_type=int,
when_used=unless
- none)],
reactions:
List[ReactionCount],
**ex-
tra_data:
Any)
```

This object represents reaction changes on a message with anonymous reactions.

Source: https://core.telegram.org/bots/api#messagereactioncountupdated

chat: Chat

The chat containing the message

message\_id: int

Unique message identifier inside the chat

## model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

 $model_post_init(context: Any, /) \rightarrow None$ 

We need to both initialize private attributes and call the user-defined model\_post\_init method.

date: DateTime

Date of the change in Unix time

reactions: List[ReactionCount]

List of reactions that are present on the message

### MessageReactionUpdated

```
class aiogram.types.message_reaction_updated.MessageReactionUpdated(*, chat: Chat, message_id:
                                                                                 int, date:
                                                                                 _datetime_serializer,
                                                                                 return_type=int,
                                                                                 when_used=unless -
                                                                                 none)], old reaction:
                                                                                 List[ReactionTypeEmoji |
                                                                                 ReactionTypeCustomEmoji
                                                                                 ReactionTypePaid],
                                                                                 new_reaction:
                                                                                 List[ReactionTypeEmoji |
                                                                                 ReactionTypeCustomEmoji
                                                                                 | ReactionTypePaid], user:
                                                                                 User | None = None,
                                                                                 actor_chat: Chat | None =
                                                                                 None, **extra_data: Any)
     This object represents a change of a reaction on a message performed by a user.
     Source: https://core.telegram.org/bots/api#messagereactionupdated
     chat:
             Chat
          The chat containing the message the user reacted to
     message_id: int
          Unique identifier of the message inside the chat
     date: DateTime
          Date of the change in Unix time
     old_reaction: List[ReactionTypeEmoji | ReactionTypeCustomEmoji | ReactionTypePaid]
          Previous list of reaction types that were set by the user
     model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
          A dictionary of computed field names and their corresponding ComputedFieldInfo objects.
     model_post_init(context: Any, /) \rightarrow None
          We need to both initialize private attributes and call the user-defined model_post_init method.
     new_reaction: List[ReactionTypeEmoji | ReactionTypeCustomEmoji | ReactionTypePaid]
          New list of reaction types that have been set by the user
     user: User | None
          Optional. The user that changed the reaction, if the user isn't anonymous
     actor_chat: Chat | None
          Optional. The chat on behalf of which the reaction was changed, if the user is anonymous
```

### **PaidMedia**

```
class aiogram.types.paid_media.PaidMedia(**extra_data: Any)
     This object describes paid media. Currently, it can be one of

    aiogram.types.paid_media_preview.PaidMediaPreview

        • aiogram.types.paid_media_photo.PaidMediaPhoto
        • aiogram.types.paid_media_video.PaidMediaVideo
     Source: https://core.telegram.org/bots/api#paidmedia
     model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
          A dictionary of computed field names and their corresponding ComputedFieldInfo objects.
     model_post_init(context: Any, /) \rightarrow None
          We need to both initialize private attributes and call the user-defined model_post_init method.
PaidMediaInfo
class aiogram.types.paid_media_info.PaidMediaInfo(*, star_count: int, paid_media:
                                                          List[PaidMediaPreview | PaidMediaPhoto |
                                                          PaidMediaVideo], **extra_data: Any)
     Describes the paid media added to a message.
     Source: https://core.telegram.org/bots/api#paidmediainfo
     star_count: int
          The number of Telegram Stars that must be paid to buy access to the media
     model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
          A dictionary of computed field names and their corresponding ComputedFieldInfo objects.
     model_post_init(context: Any, /) \rightarrow None
          We need to both initialize private attributes and call the user-defined model post init method.
     paid_media: List[PaidMediaPreview | PaidMediaPhoto | PaidMediaVideo]
          Information about the paid media
PaidMediaPhoto
class aiogram.types.paid_media_photo.PaidMediaPhoto(*, type: Literal[PaidMediaType.PHOTO] =
                                                            PaidMediaType.PHOTO, photo:
                                                            List[PhotoSize], **extra_data: Any)
     The paid media is a photo.
     Source: https://core.telegram.org/bots/api#paidmediaphoto
     type: Literal[PaidMediaType.PHOTO]
          Type of the paid media, always 'photo'
     model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
          A dictionary of computed field names and their corresponding ComputedFieldInfo objects.
```

```
model_post_init(context: Any, /) \rightarrow None
          We need to both initialize private attributes and call the user-defined model_post_init method.
     photo: List[PhotoSize]
          The photo
PaidMediaPreview
class aiogram.types.paid_media_preview.PaidMediaPreview(*, type: Literal[PaidMediaType.PREVIEW]
                                                                   = PaidMediaType.PREVIEW, width: int |
                                                                   None = None, height: int | None = None,
                                                                   duration: int \mid None = None, **extra_data:
                                                                   Any)
     The paid media isn't available before the payment.
     Source: https://core.telegram.org/bots/api#paidmediapreview
     type: Literal[PaidMediaType.PREVIEW]
          Type of the paid media, always 'preview'
     width: int | None
          Optional. Media width as defined by the sender
     model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
          A dictionary of computed field names and their corresponding ComputedFieldInfo objects.
     model_post_init(context: Any, /) \rightarrow None
          We need to both initialize private attributes and call the user-defined model_post_init method.
     height: int | None
          Optional. Media height as defined by the sender
     duration: int | None
          Optional. Duration of the media in seconds as defined by the sender
PaidMediaVideo
class aiogram.types.paid_media_video.PaidMediaVideo(*, type: Literal[PaidMediaType.VIDEO] =
                                                              PaidMediaType.VIDEO, video: Video,
                                                              **extra data: Any)
     The paid media is a video.
     Source: https://core.telegram.org/bots/api#paidmediavideo
     type: Literal[PaidMediaType.VIDE0]
          Type of the paid media, always 'video'
     model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
          A dictionary of computed field names and their corresponding ComputedFieldInfo objects.
     model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

video: Video
The video

### **PhotoSize**

```
class aiogram.types.photo_size.PhotoSize(*, file_id: str, file_unique_id: str, width: int, height: int,
                                                  file_size: int | None = None, **extra_data: Any)
     This object represents one size of a photo or a file / aiogram.methods.sticker.Sticker thumbnail.
     Source: https://core.telegram.org/bots/api#photosize
     file_id: str
           Identifier for this file, which can be used to download or reuse the file
     file_unique_id: str
           Unique identifier for this file, which is supposed to be the same over time and for different bots. Can't be
           used to download or reuse the file.
     width: int
           Photo width
     model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
           A dictionary of computed field names and their corresponding ComputedFieldInfo objects.
     model_post_init(context: Any, /) \rightarrow None
           We need to both initialize private attributes and call the user-defined model_post_init method.
     height: int
           Photo height
     file_size: int | None
           Optional. File size in bytes
Poll
class aiogram.types.poll.Poll(*, id: str, question: str, options: List[PollOption], total_voter_count: int,
                                     is_closed: bool, is_anonymous: bool, type: str, allows_multiple_answers:
                                     bool, question_entities: List[MessageEntity] | None = None,
                                     correct_option_id: int | None = None, explanation: str | None = None,
                                     explanation_entities: List[MessageEntity] | None = None, open_period: int |
                                     None = None, close date: datetime serializer, return type=int,
                                     when_used=unless - none)] | None = None, **extra_data: Any)
     This object contains information about a poll.
     Source: https://core.telegram.org/bots/api#poll
```

id: str

Unique poll identifier

question: str

Poll question, 1-300 characters

options: List[PollOption]

List of poll options

total\_voter\_count: int

Total number of users that voted in the poll

is\_closed: bool

True, if the poll is closed

is\_anonymous: bool

True, if the poll is anonymous

type: str

Poll type, currently can be 'regular' or 'quiz'

model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

 $model_post_init(context: Any, /) \rightarrow None$ 

We need to both initialize private attributes and call the user-defined model\_post\_init method.

allows\_multiple\_answers: bool

True, if the poll allows multiple answers

question\_entities: List[MessageEntity] | None

*Optional*. Special entities that appear in the *question*. Currently, only custom emoji entities are allowed in poll questions

correct\_option\_id: int | None

*Optional*. 0-based identifier of the correct answer option. Available only for polls in the quiz mode, which are closed, or was sent (not forwarded) by the bot or to the private chat with the bot.

explanation: str | None

*Optional*. Text that is shown when a user chooses an incorrect answer or taps on the lamp icon in a quiz-style poll, 0-200 characters

explanation\_entities: List[MessageEntity] | None

Optional. Special entities like usernames, URLs, bot commands, etc. that appear in the explanation

open\_period: int | None

Optional. Amount of time in seconds the poll will be active after creation

close\_date: DateTime | None

Optional. Point in time (Unix timestamp) when the poll will be automatically closed

### **PollAnswer**

**class** aiogram.types.poll\_answer.**PollAnswer**(\*, poll\_id: str, option\_ids: List[int], voter\_chat: Chat | None = None, user: User | None = None, \*\*extra data: Any)

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

Source: https://core.telegram.org/bots/api#pollanswer

poll\_id: str

Unique poll identifier

option\_ids: List[int]

0-based identifiers of chosen answer options. May be empty if the vote was retracted.

model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
voter_chat: Chat | None
```

Optional. The chat that changed the answer to the poll, if the voter is anonymous

user: *User* | None

Optional. The user that changed the answer to the poll, if the voter isn't anonymous

## **PollOption**

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

Source: https://core.telegram.org/bots/api#polloption

```
text: str
```

Option text, 1-100 characters

voter\_count: int

Number of users that voted for this option

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
text_entities: List[MessageEntity] | None
```

*Optional*. Special entities that appear in the option *text*. Currently, only custom emoji entities are allowed in poll option texts

## **ProximityAlertTriggered**

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

Source: https://core.telegram.org/bots/api#proximityalerttriggered

traveler: User

User that triggered the alert

watcher: User

User that set the alert

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model post init method.

### distance: int

The distance between the users

#### ReactionCount

Represents a reaction added to a message along with the number of times it was added.

Source: https://core.telegram.org/bots/api#reactioncount

```
type: ReactionTypeEmoji | ReactionTypeCustomEmoji | ReactionTypePaid
Type of the reaction
```

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
total_count: int
```

Number of times the reaction was added

## ReactionType

```
class aiogram.types.reaction_type.ReactionType(**extra_data: Any)
```

This object describes the type of a reaction. Currently, it can be one of

- aiogram.types.reaction\_type\_emoji.ReactionTypeEmoji
- aiogram.types.reaction\_type\_custom\_emoji.ReactionTypeCustomEmoji
- aiogram.types.reaction\_type\_paid.ReactionTypePaid

Source: https://core.telegram.org/bots/api#reactiontype

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

## ReactionTypeCustomEmoji

The reaction is based on a custom emoji.

Source: https://core.telegram.org/bots/api#reactiontypecustomemoji

## type: Literal[ReactionTypeType.CUSTOM\_EMOJI]

Type of the reaction, always 'custom\_emoji'

## model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
custom_emoji_id: str
```

Custom emoji identifier

## ReactionTypeEmoji

```
class aiogram.types.reaction_type_emoji.ReactionTypeEmoji(*, type:
```

Literal[ReactionTypeType.EMOJI] =
ReactionTypeType.EMOJI, emoji: str,
\*\*extra data: Any)

The reaction is based on an emoji.

Source: https://core.telegram.org/bots/api#reactiontypeemoji

## type: Literal[ReactionTypeType.EMOJI]

Type of the reaction, always 'emoji'

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
emoji: str
```

### ReactionTypePaid

The reaction is paid.

Source: https://core.telegram.org/bots/api#reactiontypepaid

```
type: Literal['paid']
```

Type of the reaction, always 'paid'

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

### ReplyKeyboardMarkup

class aiogram.types.reply\_keyboard\_markup.ReplyKeyboardMarkup(\*, keyboard:

List[List[KeyboardButton]],
is\_persistent: bool | None = None,
resize\_keyboard: bool | None =
None, one\_time\_keyboard: bool |
None = None,
input\_field\_placeholder: str | None
= None, selective: bool | None =
None, \*\*extra\_data: Any)

This object represents a custom keyboard with reply options (see Introduction to bots for details and examples). Not supported in channels and for messages sent on behalf of a Telegram Business account.

Source: https://core.telegram.org/bots/api#replykeyboardmarkup

## keyboard: List[List[KeyboardButton]]

Array of button rows, each represented by an Array of aiogram.types.keyboard\_button. KeyboardButton objects

### is\_persistent: bool | None

*Optional*. Requests clients to always show the keyboard when the regular keyboard is hidden. Defaults to *false*, in which case the custom keyboard can be hidden and opened with a keyboard icon.

## resize\_keyboard: bool | None

*Optional*. Requests clients to resize the keyboard vertically for optimal fit (e.g., make the keyboard smaller if there are just two rows of buttons). Defaults to *false*, in which case the custom keyboard is always of the same height as the app's standard keyboard.

## model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

### $model_post_init(context: Any, /) \rightarrow None$

We need to both initialize private attributes and call the user-defined model\_post\_init method.

## one\_time\_keyboard: bool | None

*Optional*. Requests clients to hide the keyboard as soon as it's been used. The keyboard will still be available, but clients will automatically display the usual letter-keyboard in the chat - the user can press a special button in the input field to see the custom keyboard again. Defaults to *false*.

### input\_field\_placeholder: str | None

Optional. The placeholder to be shown in the input field when the keyboard is active; 1-64 characters

### selective: bool | None

Optional. Use this parameter if you want to show the keyboard to specific users only. Targets: 1) users that are @mentioned in the *text* of the *aiogram.types.message.Message* object; 2) if the bot's message is a reply to a message in the same chat and forum topic, sender of the original message.

## ReplyKeyboardRemove

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 <code>aiogram.types.reply\_keyboard\_markup.ReplyKeyboardMarkup</code>). Not supported in channels and for messages sent on behalf of a Telegram Business account.

Source: https://core.telegram.org/bots/api#replykeyboardremove

## remove\_keyboard: Literal[True]

Requests clients to remove the custom keyboard (user will not be able to summon this keyboard; if you want to hide the keyboard from sight but keep it accessible, use *one\_time\_keyboard* in *aiogram.types.reply\_keyboard\_markup*.ReplyKeyboardMarkup)

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
selective: bool | None
```

*Optional*. Use this parameter if you want to remove the keyboard for specific users only. Targets: 1) users that are @mentioned in the *text* of the *aiogram.types.message.Message* object; 2) if the bot's message is a reply to a message in the same chat and forum topic, sender of the original message.

## ReplyParameters

```
class aiogram.types.reply_parameters.ReplyParameters(*, message_id: int, chat_id: int | str | None = None, allow_sending_without_reply: bool | ~aiogram.client.default.Default | None = < Default('allow_sending_without_reply')>, quote: str | None = None, quote_parse_mode: str | ~aiogram.client.default.Default | None = < Default('parse_mode')>, quote_entities: ~typing.List[~aiogram.types.message_entity.MessageEntity] | None = None, quote_position: int | None = None, **extra_data: ~typing.Any)
```

Describes reply parameters for the message that is being sent.

Source: https://core.telegram.org/bots/api#replyparameters

### message\_id: int

Identifier of the message that will be replied to in the current chat, or in the chat *chat\_id* if it is specified

```
chat_id: int | str | None
```

*Optional*. If the message to be replied to is from a different chat, unique identifier for the chat or username of the channel (in the format @channelusername). Not supported for messages sent on behalf of a business account.

## allow\_sending\_without\_reply: bool | Default | None

*Optional*. Pass True if the message should be sent even if the specified message to be replied to is not found. Always False for replies in another chat or forum topic. Always True for messages sent on behalf of a business account.

### quote: str | None

*Optional.* Quoted part of the message to be replied to; 0-1024 characters after entities parsing. The quote must be an exact substring of the message to be replied to, including *bold*, *italic*, *underline*, *strikethrough*, *spoiler*, and *custom\_emoji* entities. The message will fail to send if the quote isn't found in the original message.

## model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
quote_parse_mode: str | Default | None
```

Optional. Mode for parsing entities in the quote. See formatting options for more details.

```
quote_entities: List[MessageEntity] | None
```

*Optional.* A JSON-serialized list of special entities that appear in the quote. It can be specified instead of *quote\_parse\_mode*.

```
quote_position: int | None
```

Optional. Position of the quote in the original message in UTF-16 code units

## ResponseParameters

Describes why a request was unsuccessful.

Source: https://core.telegram.org/bots/api#responseparameters

```
migrate_to_chat_id: int | None
```

*Optional*. The group has been migrated to a supergroup with the specified identifier. This number may have more than 32 significant bits and some programming languages may have difficulty/silent defects in interpreting it. But it has at most 52 significant bits, so a signed 64-bit integer or double-precision float type are safe for storing this identifier.

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
retry_after: int | None
```

272

Optional. In case of exceeding flood control, the number of seconds left to wait before the request can be repeated

### **SharedUser**

```
class aiogram.types.shared_user.SharedUser(*, user_id: int, first_name: str \mid None = None, last_name: str \mid None = None, username: str \mid None = None, photo: List[PhotoSize] \mid None = None, **extra_data: Any)
```

This object contains information about a user that was shared with the bot using a aiogram.types. keyboard\_button\_request\_users.KeyboardButtonRequestUsers button.

Source: https://core.telegram.org/bots/api#shareduser

```
user_id: int
```

Identifier of the shared user. This number may have more than 32 significant bits and some programming languages may have difficulty/silent defects in interpreting it. But it has at most 52 significant bits, so 64-bit integers or double-precision float types are safe for storing these identifiers. The bot may not have access to the user and could be unable to use this identifier, unless the user is already known to the bot by some other means.

```
first_name: str | None
```

Optional. First name of the user, if the name was requested by the bot

```
last_name: str | None
```

Optional. Last name of the user, if the name was requested by the bot

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
username: str | None
```

Optional. Username of the user, if the username was requested by the bot

```
photo: List[PhotoSize] | None
```

Optional. Available sizes of the chat photo, if the photo was requested by the bot

### Story

```
class aiogram.types.story.Story(*, chat: Chat, id: int, **extra_data: Any)
```

This object represents a story.

Source: https://core.telegram.org/bots/api#story

chat: Chat

Chat that posted the story

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

id: int

Unique identifier for the story in the chat

# SwitchInlineQueryChosenChat

```
class aiogram.types.switch_inline_query_chosen_chat.SwitchInlineQueryChosenChat(*, query: str
                                                                                              | None =
                                                                                              None, al-
                                                                                              low_user_chats:
                                                                                              bool | None
                                                                                              = None, al-
                                                                                              low bot chats:
                                                                                              bool | None
                                                                                              = None, al-
                                                                                              low_group_chats:
                                                                                              bool | None
                                                                                              = None, al-
                                                                                              low channel chats:
                                                                                              bool | None
                                                                                              = None,
                                                                                              **ex-
                                                                                              tra_data:
                                                                                              Any)
```

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

Source: https://core.telegram.org/bots/api#switchinlinequerychosenchat

```
query: str | None
```

Optional. The default inline query to be inserted in the input field. If left empty, only the bot's username will be inserted

```
allow_user_chats: bool | None
```

Optional. True, if private chats with users can be chosen

```
allow_bot_chats: bool | None
```

Optional. True, if private chats with bots can be chosen

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
allow_group_chats: bool | None
```

Optional. True, if group and supergroup chats can be chosen

```
allow_channel_chats: bool | None
```

Optional. True, if channel chats can be chosen

### **TextQuote**

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

Source: https://core.telegram.org/bots/api#textquote

text: str

Text of the quoted part of a message that is replied to by the given message

position: int

Approximate quote position in the original message in UTF-16 code units as specified by the sender

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
entities: List[MessageEntity] | None
```

*Optional*. Special entities that appear in the quote. Currently, only *bold*, *italic*, *underline*, *strikethrough*, *spoiler*, and *custom\_emoji* entities are kept in quotes.

```
is_manual: bool | None
```

*Optional*. True, if the quote was chosen manually by the message sender. Otherwise, the quote was added automatically by the server.

## User

This object represents a Telegram user or bot.

Source: https://core.telegram.org/bots/api#user

#### id: int

Unique identifier for this user or bot. This number may have more than 32 significant bits and some programming languages may have difficulty/silent defects in interpreting it. But it has at most 52 significant bits, so a 64-bit integer or double-precision float type are safe for storing this identifier.

is\_bot: bool

True, if this user is a bot

first\_name: str

User's or bot's first name

last\_name: str | None

Optional. User's or bot's last name

```
username: str | None
     Optional. User's or bot's username
language_code: str | None
     Optional. IETF language tag of the user's language
is_premium: bool | None
     Optional. True, if this user is a Telegram Premium user
added_to_attachment_menu: bool | None
     Optional. True, if this user added the bot to the attachment menu
can_join_groups: bool | None
     Optional. True, if the bot can be invited to groups. Returned only in aiogram.methods.get_me.GetMe.
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
     A dictionary of computed field names and their corresponding ComputedFieldInfo objects.
model_post_init(context: Any, /) \rightarrow None
     We need to both initialize private attributes and call the user-defined model_post_init method.
can_read_all_group_messages: bool | None
     Optional. True, if privacy mode is disabled for the bot. Returned only in aiogram.methods.get_me.
     GetMe.
supports_inline_queries: bool | None
     Optional. True, if the bot supports inline queries. Returned only in aiogram.methods.get_me.GetMe.
can_connect_to_business: bool | None
     Optional. True, if the bot can be connected to a Telegram Business account to receive its messages. Re-
     turned only in aiogram.methods.get_me.GetMe.
has_main_web_app: bool | None
     Optional. True, if the bot has a main Web App. Returned only in aiogram.methods.get_me.GetMe.
property full_name: str
property url: str
mention_markdown(name: str \mid None = None) \rightarrow str
mention_html (name: str \mid None = None) \rightarrow str
get_profile_photos(offset: int | None = None, limit: int | None = None, **kwargs: Any) \rightarrow
                      GetUserProfilePhotos
     Shortcut for method aiogram.methods.get_user_profile_photos.GetUserProfilePhotos will
     automatically fill method attributes:
       • user_id
     Use this method to get a list of profile pictures for a user.
                                                                        Returns a aiogram.types.
     user_profile_photos.UserProfilePhotos object.
```

**Parameters** 

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

• **offset** – Sequential number of the first photo to be returned. By default, all photos are returned.

• limit – Limits the number of photos to be retrieved. Values between 1-100 are accepted. Defaults to 100.

#### Returns

```
\begin{array}{lll} \text{instance} & \text{of} & \text{method} & \textit{aiogram.methods.get\_user\_profile\_photos}. \\ \textit{GetUserProfilePhotos} & \end{array}
```

#### **UserChatBoosts**

```
class aiogram.types.user_chat_boosts.UserChatBoosts(*, boosts: List[ChatBoost], **extra_data: Any)
This object represents a list of boosts added to a chat by a user.
```

Source: https://core.telegram.org/bots/api#userchatboosts

boosts: List[ChatBoost]

The list of boosts added to the chat by the user

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

 $model_post_init(context: Any, /) \rightarrow None$ 

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### **UserProfilePhotos**

This object represent a user's profile pictures.

Source: https://core.telegram.org/bots/api#userprofilephotos

total\_count: int

Total number of profile pictures the target user has

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

photos: List[List[PhotoSize]]

Requested profile pictures (in up to 4 sizes each)

#### **UserShared**

```
class aiogram.types.user_shared.UserShared(*, request_id: int, user_id: int, **extra_data: Any)
```

This object contains information about the user whose identifier was shared with the bot using a aiogram. types.keyboard\_button\_request\_user.KeyboardButtonRequestUser button.

Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023

Source: https://core.telegram.org/bots/api#usershared

#### request\_id: int

Identifier of the request

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### user\_id: int

Identifier of the shared user. This number may have more than 32 significant bits and some programming languages may have difficulty/silent defects in interpreting it. But it has at most 52 significant bits, so a 64-bit integer or double-precision float type are safe for storing this identifier. The bot may not have access to the user and could be unable to use this identifier, unless the user is already known to the bot by some other means.

#### **UsersShared**

This object contains information about the users whose identifiers were shared with the bot using a aiogram. types.keyboard\_button\_request\_users.KeyboardButtonRequestUsers button.

Source: https://core.telegram.org/bots/api#usersshared

### request\_id: int

Identifier of the request

#### users: List[SharedUser]

Information about users shared with the bot.

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
user_ids: List[int] | None
```

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

Deprecated since version API:7.2: https://core.telegram.org/bots/api-changelog#march-31-2024

### Venue

```
class aiogram.types.venue.Venue(*, location: Location, title: str, address: str, foursquare_id: str | None =
                                      None, foursquare_type: str | None = None, google_place_id: str | None =
                                      None, google_place_type: str | None = None, **extra_data: Any)
     This object represents a venue.
     Source: https://core.telegram.org/bots/api#venue
     location: Location
          Venue location. Can't be a live location
     title: str
          Name of the venue
     address: str
          Address of the venue
     foursquare_id: str | None
          Optional. Foursquare identifier of the venue
     model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
          A dictionary of computed field names and their corresponding ComputedFieldInfo objects.
     model_post_init(context: Any, /) \rightarrow None
          We need to both initialize private attributes and call the user-defined model post init method.
     foursquare_type: str | None
          Optional.
                         Foursquare type of the venue.
                                                                 (For example,
                                                                                  'arts_entertainment/default',
          'arts_entertainment/aquarium' or 'food/icecream'.)
     google_place_id: str | None
          Optional. Google Places identifier of the venue
     google_place_type: str | None
          Optional. Google Places type of the venue. (See supported types.)
```

### **Video**

This object represents a video file.

Source: https://core.telegram.org/bots/api#video

### file\_id: str

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

### file\_unique\_id: str

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

#### width: int

Video width as defined by the sender

height: int

Video height as defined by the sender

duration: int

Duration of the video in seconds as defined by the sender

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

thumbnail: PhotoSize | None

Optional. Video thumbnail

file\_name: str | None

Optional. Original filename as defined by the sender

mime\_type: str | None

Optional. MIME type of the file as defined by the sender

file\_size: int | None

*Optional.* File size in bytes. It can be bigger than 2^31 and some programming languages may have difficulty/silent defects in interpreting it. But it has at most 52 significant bits, so a signed 64-bit integer or double-precision float type are safe for storing this value.

#### VideoChatEnded

class aiogram.types.video\_chat\_ended.VideoChatEnded(\*, duration: int, \*\*extra\_data: Any)

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

Source: https://core.telegram.org/bots/api#videochatended

duration: int

Video chat duration in seconds

model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

 $model_post_init(context: Any, /) \rightarrow None$ 

We need to both initialize private attributes and call the user-defined model\_post\_init method.

## VideoChatParticipantsInvited

class aiogram.types.video\_chat\_participants\_invited.VideoChatParticipantsInvited(\*, users:

```
List[User],
**ex-
tra_data:
```

Any)

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

Source: https://core.telegram.org/bots/api#videochatparticipantsinvited

users: List[User]

New members that were invited to the video chat

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### VideoChatScheduled

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

Source: https://core.telegram.org/bots/api#videochatscheduled

```
start_date: DateTime
```

Point in time (Unix timestamp) when the video chat is supposed to be started by a chat administrator

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model post init method.

### VideoChatStarted

```
class aiogram.types.video_chat_started.VideoChatStarted(**extra data: Any)
```

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

Source: https://core.telegram.org/bots/api#videochatstarted

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

# VideoNote

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

Source: https://core.telegram.org/bots/api#videonote

```
file_id: str
```

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

# file\_unique\_id: str

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

### length: int

Video width and height (diameter of the video message) as defined by the sender

# model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

### $model_post_init(context: Any, /) \rightarrow None$

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### duration: int

Duration of the video in seconds as defined by the sender

## thumbnail: PhotoSize | None

Optional. Video thumbnail

## file\_size: int | None

Optional. File size in bytes

### Voice

This object represents a voice note.

Source: https://core.telegram.org/bots/api#voice

#### file\_id: str

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

### file\_unique\_id: str

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

#### duration: int

Duration of the audio in seconds as defined by the sender

### model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### mime\_type: str | None

Optional. MIME type of the file as defined by the sender

### file\_size: int | None

*Optional.* File size in bytes. It can be bigger than 2^31 and some programming languages may have difficulty/silent defects in interpreting it. But it has at most 52 significant bits, so a signed 64-bit integer or double-precision float type are safe for storing this value.

## WebAppData

class aiogram.types.web\_app\_data.WebAppData(\*, data: str, button\_text: str, \*\*extra\_data: Any)

Describes data sent from a Web App to the bot.

Source: https://core.telegram.org/bots/api#webappdata

data: str

The data. Be aware that a bad client can send arbitrary data in this field.

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
button_text: str
```

Text of the web\_app keyboard button from which the Web App was opened. Be aware that a bad client can send arbitrary data in this field.

## WebAppInfo

class aiogram.types.web\_app\_info.WebAppInfo(\*, url: str, \*\*extra\_data: Any)

Describes a Web App.

Source: https://core.telegram.org/bots/api#webappinfo

url: str

An HTTPS URL of a Web App to be opened with additional data as specified in Initializing Web Apps

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### WriteAccessAllowed

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

Source: https://core.telegram.org/bots/api#writeaccessallowed

```
from_request: bool | None
```

*Optional*. True, if the access was granted after the user accepted an explicit request from a Web App sent by the method requestWriteAccess

```
web_app_name: str | None
```

Optional. Name of the Web App, if the access was granted when the Web App was launched from a link

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
from_attachment_menu: bool | None
```

Optional. True, if the access was granted when the bot was added to the attachment or side menu

#### Inline mode

#### ChosenInlineResult

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.

Source: https://core.telegram.org/bots/api#choseninlineresult

```
result_id: str
```

The unique identifier for the result that was chosen

from\_user: User

The user that chose the result

query: str

The query that was used to obtain the result

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
location: Location | None
```

Optional. Sender location, only for bots that require user location

```
inline_message_id: str | None
```

*Optional*. Identifier of the sent inline message. Available only if there is an inline keyboard attached to the message. Will be also received in callback queries and can be used to edit the message.

## InlineQuery

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

Source: https://core.telegram.org/bots/api#inlinequery

#### id: str

Unique identifier for this query

from\_user: User

Sender query: str

Text of the query (up to 256 characters)

offset: str

Offset of the results to be returned, can be controlled by the bot

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### chat\_type: str | None

*Optional*. Type of the chat from which the inline query was sent. Can be either 'sender' for a private chat with the inline query sender, 'private', 'group', 'supergroup', or 'channel'. The chat type should be always known for requests sent from official clients and most third-party clients, unless the request was sent from a secret chat

### location: Location | None

Optional. Sender location, only for bots that request user location

answer(results: List[InlineQueryResultCachedAudio | InlineQueryResultCachedDocument |

InlineQueryResultCachedGif | InlineQueryResultCachedMpeg4Gif | InlineQueryResultCachedPhoto | InlineQueryResultCachedSticker | InlineQueryResultCachedVideo | InlineQueryResultCachedVoice

| InlineQueryResultArticle | InlineQueryResultAudio | InlineQueryResultContact |

InlineQueryResultGame | InlineQueryResultDocument | InlineQueryResultGif |

InlineQueryResultLocation | InlineQueryResultMpeg4Gif | InlineQueryResultPhoto |

InlineQueryResultVenue | InlineQueryResultVideo | InlineQueryResultVoice], cache\_time: int |

None = None, is personal: bool | None = None, next offset:  $str \mid None = None$ , button:

InlineQueryResultsButton | None = None,  $switch\_pm\_parameter$ :  $str \mid None = None$ ,  $switch\_pm\_text$ :  $str \mid None = None$ , \*\*kwargs: Any)  $\rightarrow AnswerInlineOuery$ 

Shortcut for method aiogram.methods.answer\_inline\_query.AnswerInlineQuery will automatically fill method attributes:

inline\_query\_id

Use this method to send answers to an inline query. On success, True is returned.

No more than 50 results per query are allowed.

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

#### **Parameters**

- results A JSON-serialized array of results for the inline query
- **cache\_time** 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** 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** 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.

- button A JSON-serialized object describing a button to be shown above inline query results
- **switch\_pm\_parameter** 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.
- **switch\_pm\_text** 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*

#### **Returns**

instance of method aiogram.methods.answer\_inline\_query.AnswerInlineQuery

# InlineQueryResult

class aiogram.types.inline\_query\_result.InlineQueryResult(\*\*extra\_data: Any)

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

- aiogram.types.inline\_query\_result\_cached\_audio.InlineQueryResultCachedAudio
- aiogram.types.inline\_query\_result\_cached\_document.InlineQueryResultCachedDocument
- aiogram.types.inline\_query\_result\_cached\_gif.InlineQueryResultCachedGif
- $\bullet \ aiogram.\ types.inline\_query\_result\_cached\_mpeg4\_gif. In lineQueryResultCachedMpeg4Gif$
- aiogram.types.inline\_query\_result\_cached\_photo.InlineQueryResultCachedPhoto
- aiogram.types.inline\_query\_result\_cached\_sticker.InlineQueryResultCachedSticker
- aiogram.types.inline\_query\_result\_cached\_video.InlineQueryResultCachedVideo
- aiogram.types.inline\_query\_result\_cached\_voice.InlineQueryResultCachedVoice
- aiogram.types.inline\_query\_result\_article.InlineQueryResultArticle
- aiogram.types.inline\_query\_result\_audio.InlineQueryResultAudio
- aiogram.types.inline\_query\_result\_contact.InlineQueryResultContact
- aiogram.types.inline\_query\_result\_game.InlineQueryResultGame
- aiogram.types.inline\_query\_result\_document.InlineQueryResultDocument
- aiogram.types.inline\_query\_result\_gif.InlineQueryResultGif
- aiogram.types.inline\_query\_result\_location.InlineQueryResultLocation
- aiogram.types.inline\_query\_result\_mpeg4\_gif.InlineQueryResultMpeg4Gif
- aiogram.types.inline\_query\_result\_photo.InlineQueryResultPhoto
- aiogram.types.inline\_query\_result\_venue.InlineQueryResultVenue
- aiogram.types.inline\_query\_result\_video.InlineQueryResultVideo
- aiogram.types.inline\_query\_result\_voice.InlineQueryResultVoice

**Note:** All URLs passed in inline query results will be available to end users and therefore must be assumed to be **public**.

Source: https://core.telegram.org/bots/api#inlinequeryresult

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

## **InlineQueryResultArticle**

```
class aiogram.types.inline_query_result_article.InlineQueryResultArticle(*, type: Lit-
```

eral[InlineQueryResultType.ARTICLE] = Inline Query Result-Type.ARTICLE, id: str, title: str, input message content: InputTextMessage-Content | InputLocationMessageContent | InputVenueMessage-Content | InputContactMessageContent | InputInvoiceMessageContent, reply\_markup: InlineKeyboardMarkup | None = None, url: $str \mid None = None,$ hide url: bool | None = *None*, description:  $str \mid None = None,$ thumbnail\_url: str | None = None,thumbnail\_width: int | None = None,thumbnail\_height:  $int \mid None = None,$ \*\*extra\_data: Any)

Represents a link to an article or web page.

Source: https://core.telegram.org/bots/api#inlinequeryresultarticle

## type: Literal[InlineQueryResultType.ARTICLE]

Type of the result, must be article

## id: str

Unique identifier for this result, 1-64 Bytes

# title: str

Title of the result

input\_message\_content: InputTextMessageContent | InputLocationMessageContent |
InputVenueMessageContent | InputContactMessageContent | InputInvoiceMessageContent

Content of the message to be sent

```
reply_markup: InlineKeyboardMarkup | None
     Optional. Inline keyboard attached to the message
url: str | None
     Optional. URL of the result
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
     A dictionary of computed field names and their corresponding ComputedFieldInfo objects.
model_post_init(context: Any, /) \rightarrow None
     We need to both initialize private attributes and call the user-defined model_post_init method.
hide_url: bool | None
     Optional. Pass True if you don't want the URL to be shown in the message
description: str | None
     Optional. Short description of the result
thumbnail_url: str | None
     Optional. Url of the thumbnail for the result
thumbnail_width: int | None
     Optional. Thumbnail width
thumbnail_height: int | None
     Optional. Thumbnail height
```

InlineQueryResultAudio

= InlineQueryResult-Type.AUDIO, id: str, audio\_url: str, title: str, caption: str | None = None, parse mode: str |

ing.Literal[InlineQueryResultType.AUDIO]

```
~aiogram.client.default.Default
                                                                               | None =
                                                                               <Default('parse_mode')>,
                                                                               caption_entities: ~typ-
                                                                               ing.List[~aiogram.types.message_entity.Mes.
                                                                               | None = None, performer:
                                                                               str \mid None = None,
                                                                               audio_duration: int | None
                                                                               = None, reply_markup:
                                                                               ~aiogram.types.inline_keyboard_markup.Inli
                                                                               | None = None,
                                                                               input_message_content:
                                                                               ~aiogram.types.input_text_message_content.
                                                                               ~aiogram.types.input_location_message_com
                                                                               ~aiogram.types.input_venue_message_conten
                                                                               ~aiogram.types.input_contact_message_cont
                                                                               ~aiogram.types.input_invoice_message_cont
                                                                               | None = None,
                                                                                **extra data:
                                                                               ~typing.Any)
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.
Source: https://core.telegram.org/bots/api#inlinequeryresultaudio
type: Literal[InlineQueryResultType.AUDIO]
     Type of the result, must be audio
id: str
     Unique identifier for this result, 1-64 bytes
audio url: str
     A valid URL for the audio file
title: str
     Title
caption: str | None
     Optional. Caption, 0-1024 characters after entities parsing
parse_mode: str | Default | None
     Optional. Mode for parsing entities in the audio caption. See formatting options for more details.
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
     A dictionary of computed field names and their corresponding ComputedFieldInfo objects.
```

class aiogram.types.inline\_query\_result\_audio.InlineQueryResultAudio(\*, type: ~typ-

```
model_post_init(context: Any, /) → None
    We need to both initialize private attributes and call the user-defined model_post_init method.

caption_entities: List[MessageEntity] | None
    Optional. List of special entities that appear in the caption, which can be specified instead of parse_mode

performer: str | None
    Optional. Performer

audio_duration: int | None
    Optional. Audio duration in seconds

reply_markup: InlineKeyboardMarkup | None
    Optional. Inline keyboard attached to the message

input_message_content: InputTextMessageContent | InputLocationMessageContent |
InputVenueMessageContent | InputContactMessageContent | InputInvoiceMessageContent |
None
```

Optional. Content of the message to be sent instead of the audio

## InlineQueryResultCachedAudio

```
ing.Literal[InlineQueryRest
= Inline-
QueryRe-
sult-
Type.AUDIO,
id: str,
au-
dio_file_id:
str,
caption:
str | None
= None,
parse_mode:
str |
~aiogram.client.default.Def
|None|
<De-
fault('parse_mode')>,
cap-
tion_entities:
~typ-
ing.List[~aiogram.types.me.
|None|
None, re-
ply_markup:
~aiogram.types.inline_keyb
|None|
None, in-
put_message_content:
~aiogram.types.input_text_r
~aiogram.types.input_locat
~aiogram.types.input_venue
~aiogram.types.input_conta
~aiogram.types.input_invoid
|None|
None,
**ex-
tra_data:
~typ-
ing.Any)
```

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 <code>input\_message\_content</code> to send a message with the specified content instead of the audio.

Source: https://core.telegram.org/bots/api#inlinequeryresultcachedaudio

# type: Literal[InlineQueryResultType.AUDIO]

Type of the result, must be audio

```
id: str
     Unique identifier for this result, 1-64 bytes
audio_file_id: str
     A valid file identifier for the audio file
caption: str | None
     Optional. Caption, 0-1024 characters after entities parsing
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
     A dictionary of computed field names and their corresponding ComputedFieldInfo objects.
model_post_init(context: Any, /) \rightarrow None
     We need to both initialize private attributes and call the user-defined model_post_init method.
parse_mode: str | Default | None
     Optional. Mode for parsing entities in the audio caption. See formatting options for more details.
caption_entities: List[MessageEntity] | None
     Optional. List of special entities that appear in the caption, which can be specified instead of parse_mode
reply_markup: InlineKeyboardMarkup | None
     Optional. Inline keyboard attached to the message
input_message_content: InputTextMessageContent | InputLocationMessageContent |
InputVenueMessageContent | InputContactMessageContent | InputInvoiceMessageContent |
None
```

Optional. Content of the message to be sent instead of the audio

InlineQueryResultCachedDocument

```
{\bf class}\ {\bf aiogram.types.inline\_query\_result\_cached\_document.} {\bf InlineQueryResultCachedDocument} (*, {\bf class}\ {\bf
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        type:
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        ing.Literal[InlineQu
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        In-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        line-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        QueryRe-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        sult-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         Type.DOCUMENT,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        str,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        ti-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        tle:
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        str,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        doc-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        u-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      ment_file_id:
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      str,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        de-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      scrip-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        tion:
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        str
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        None
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        None,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        cap-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        tion:
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        str
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        None
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      None,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      parse_mode:
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      str
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         ~aiogram.client.defe
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        None
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         <De-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     fault('parse_mode'):
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        cap-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        tion\_entities:
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        ~typ-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        ing.List[~aiogram.t
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        None
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        None,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      ply_markup:
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        ~aiogram.types.inlin
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        None
2.3. Bot API
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           293
```

put\_message\_conten

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.

Source: https://core.telegram.org/bots/api#inlinequeryresultcacheddocument

#### type: Literal[InlineQueryResultType.DOCUMENT]

Type of the result, must be document

#### id: str

Unique identifier for this result, 1-64 bytes

#### title: str

Title for the result

## document\_file\_id: str

A valid file identifier for the file

### description: str | None

Optional. Short description of the result

# model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

## $model_post_init(context: Any, /) \rightarrow None$

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### caption: str | None

Optional. Caption of the document to be sent, 0-1024 characters after entities parsing

## parse\_mode: str | Default | None

Optional. Mode for parsing entities in the document caption. See formatting options for more details.

### caption\_entities: List[MessageEntity] | None

Optional. List of special entities that appear in the caption, which can be specified instead of parse\_mode

#### reply\_markup: InlineKeyboardMarkup | None

Optional. Inline keyboard attached to the message

```
input_message_content: InputTextMessageContent | InputLocationMessageContent |
InputVenueMessageContent | InputContactMessageContent | InputInvoiceMessageContent |
None
```

Optional. Content of the message to be sent instead of the file

## InlineQueryResultCachedGif

```
class aiogram.types.inline_query_result_cached_gif.InlineQueryResultCachedGif(*, type: ~typ-
                                                                                                ing.Literal[InlineQueryResultTyp
                                                                                                = Inline-
                                                                                                QueryResult-
                                                                                                 Type.GIF, id:
                                                                                                str, gif_file_id:
                                                                                                str, title: str |
                                                                                                None = None,
                                                                                                caption: str |
                                                                                                None = None,
                                                                                                parse_mode:
                                                                                                str |
                                                                                                ~aiogram.client.default.Default
                                                                                                |None = < De-
                                                                                                fault('parse_mode')>,
                                                                                                cap-
                                                                                                tion_entities:
                                                                                                ~tvp-
                                                                                                ing.List[~aiogram.types.message
                                                                                                |None = None,
                                                                                                show\_caption\_above\_media:
                                                                                                bool |
                                                                                                ~aiogram.client.default.Default
                                                                                                |None = < De-
                                                                                                fault('show_caption_above_medi
                                                                                                reply_markup:
                                                                                                ~aiogram.types.inline_keyboard_
                                                                                                | None =
                                                                                                None, in-
                                                                                                put_message_content:
                                                                                                 ~aiogram.types.input_text_messo
                                                                                                 ~aiogram.types.input_location_n
                                                                                                ~aiogram.types.input_venue_mes
                                                                                                 ~aiogram.types.input_contact_m
                                                                                                 ~aiogram.types.input_invoice_m
                                                                                                | None =
                                                                                                None,
                                                                                                 **extra_data:
                                                                                                ~typing.Any)
```

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 <code>input\_message\_content</code> to send a message with specified content instead of the animation.

Source: https://core.telegram.org/bots/api#inlinequeryresultcachedgif

### type: Literal[InlineQueryResultType.GIF]

Type of the result, must be gif

id: str

Unique identifier for this result, 1-64 bytes

```
gif_file_id: str
     A valid file identifier for the GIF file
title: str | None
     Optional. Title for the result
caption: str | None
     Optional. Caption of the GIF file to be sent, 0-1024 characters after entities parsing
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
     A dictionary of computed field names and their corresponding ComputedFieldInfo objects.
model_post_init(context: Any, /) \rightarrow None
     We need to both initialize private attributes and call the user-defined model_post_init method.
parse_mode: str | Default | None
     Optional. Mode for parsing entities in the caption. See formatting options for more details.
caption_entities: List[MessageEntity] | None
     Optional. List of special entities that appear in the caption, which can be specified instead of parse_mode
show_caption_above_media: bool | Default | None
     Optional. Pass True, if the caption must be shown above the message media
reply_markup: InlineKeyboardMarkup | None
     Optional. Inline keyboard attached to the message
input_message_content: InputTextMessageContent | InputLocationMessageContent |
InputVenueMessageContent | InputContactMessageContent | InputInvoiceMessageContent |
None
```

Optional. Content of the message to be sent instead of the GIF animation

InlineQueryResultCachedMpeg4Gif

```
\textbf{class} \ \texttt{aiogram.types.inline\_query\_result\_cached\_mpeg4\_gif.} \\ \textbf{InlineQueryResultCachedMpeg4Gif}(*, \texttt{aiogram.types.inline\_query\_result\_cached\_mpeg4\_gif.} \\ \textbf{Aiogram.types.inline\_query\_result\_cached\_mpeg4\_gif.} \\ \textbf{Aiogram.types.types.type}(*, \texttt{aiogram.types.types.types.types.} \\ \textbf{Aiogram.types.types.types.types.types.types.types.} \\ \textbf{Aiogram.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.types.ty
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           type:
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           ing.Literal[InlineQ
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           In-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           line-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           QueryRe-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           sult-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           Type.MPEG4_GIF
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           str,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           mpeg4_file_id:
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           str,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           ti-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           tle:
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           str
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           None
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           None,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           cap-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           tion:
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           str
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           None
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          None,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           parse_mode:
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           str
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            ~aiogram.client.de
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          None
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            <De-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           fault('parse_mode'
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           cap-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           tion_entities:
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           ~typ-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           ing.List[~aiogram.
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           None
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           None,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           show_caption_abo
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           bool
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            ~aiogram.client.de
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           None
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            <De-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           fault('show_captio
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           re-
```

\_\_\_ply\_markup: **297**~aiogram.types.ini 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.

Source: https://core.telegram.org/bots/api#inlinequeryresultcachedmpeg4gif

```
type: Literal[InlineQueryResultType.MPEG4_GIF]
```

Type of the result, must be mpeg4\_gif

id: str

Unique identifier for this result, 1-64 bytes

mpeg4\_file\_id: str

A valid file identifier for the MPEG4 file

title: str | None

Optional. Title for the result

caption: str | None

Optional. Caption of the MPEG-4 file to be sent, 0-1024 characters after entities parsing

model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

 $model_post_init(context: Any, /) \rightarrow None$ 

We need to both initialize private attributes and call the user-defined model\_post\_init method.

parse\_mode: str | Default | None

Optional. Mode for parsing entities in the caption. See formatting options for more details.

caption\_entities: List[MessageEntity] | None

Optional. List of special entities that appear in the caption, which can be specified instead of parse\_mode

show\_caption\_above\_media: bool | Default | None

Optional. Pass True, if the caption must be shown above the message media

reply\_markup: InlineKeyboardMarkup | None

Optional. Inline keyboard attached to the message

input\_message\_content: InputTextMessageContent | InputLocationMessageContent |
InputVenueMessageContent | InputContactMessageContent | InputInvoiceMessageContent |
None

Optional. Content of the message to be sent instead of the video animation

## InlineQueryResultCachedPhoto

```
class aiogram.types.inline_query_result_cached_photo.InlineQueryResultCachedPhoto(*, type:
                                                                                                     ~typ-
                                                                                                     ing.Literal[InlineQueryRest
                                                                                                     = In line -
                                                                                                     QueryRe-
                                                                                                     sult-
                                                                                                     Type.PHOTO,
                                                                                                     id: str,
                                                                                                     photo_file_id:
                                                                                                     str, title:
                                                                                                     str | None
                                                                                                     = None,
                                                                                                     descrip-
                                                                                                     tion: str |
                                                                                                     None =
                                                                                                     None,
                                                                                                     caption:
                                                                                                     str | None
                                                                                                     = None,
                                                                                                     parse_mode:
                                                                                                     ~aiogram.client.default.Def
                                                                                                     | None =
                                                                                                     <De-
                                                                                                     fault('parse_mode')>,
                                                                                                     cap-
                                                                                                     tion\_entities:
                                                                                                     ~typ-
                                                                                                     ing.List[~aiogram.types.me.
                                                                                                     |None|
                                                                                                     None,
                                                                                                     show_caption_above_medic
                                                                                                     ~aiogram.client.default.Def
                                                                                                     |None|
                                                                                                     <De-
                                                                                                     fault('show_caption_above_
                                                                                                     ply_markup:
                                                                                                     ~aiogram.types.inline_keyb
                                                                                                     |None|
                                                                                                     None, in-
                                                                                                     put_message_content:
                                                                                                     ~aiogram.types.input_text_r
                                                                                                     ~aiogram.types.input_locate
                                                                                                     ~aiogram.types.input_venue
                                                                                                     ~aiogram.types.input_conta
                                                                                                     ~aiogram.types.input_invoid
                                                                                                     | None =
                                                                                                     None,
                                                                                                     **ex-
                                                                                                     tra data:
2.3. Bot API
                                                                                                     ~typ- 299
```

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.

Source: https://core.telegram.org/bots/api#inlinequeryresultcachedphoto

```
type: Literal[InlineQueryResultType.PHOT0]
```

Type of the result, must be photo

id: str

Unique identifier for this result, 1-64 bytes

photo\_file\_id: str

A valid file identifier of the photo

title: str | None

Optional. Title for the result

description: str | None

Optional. Short description of the result

caption: str | None

Optional. Caption of the photo to be sent, 0-1024 characters after entities parsing

model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

 $model_post_init(context: Any, /) \rightarrow None$ 

We need to both initialize private attributes and call the user-defined model\_post\_init method.

parse\_mode: str | Default | None

Optional. Mode for parsing entities in the photo caption. See formatting options for more details.

caption\_entities: List[MessageEntity] | None

Optional. List of special entities that appear in the caption, which can be specified instead of parse\_mode

show\_caption\_above\_media: bool | Default | None

Optional. Pass True, if the caption must be shown above the message media

reply\_markup: InlineKeyboardMarkup | None

Optional. Inline keyboard attached to the message

input\_message\_content: InputTextMessageContent | InputLocationMessageContent |
InputVenueMessageContent | InputContactMessageContent | InputInvoiceMessageContent |
None

Optional. Content of the message to be sent instead of the photo

# InlineQueryResultCachedSticker

```
{\bf class} \ {\bf aiogram.types.inline\_query\_result\_cached\_sticker.} {\bf InlineQueryResultCachedSticker} (*, {\bf class}) {\bf class} \ {\bf class
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             type:
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             eral[InlineQueryResul
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             In-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             line-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              QueryRe-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             sult-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             Type.STICKER,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             id:
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             str,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             sticker_file_id:
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             str,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              re-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             ply_markup:
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             In-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             lineKey-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             board-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             Markup
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             None
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             None,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             in-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             put_message_content:
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             put-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             TextMes-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             sage-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             Con-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             tent
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             In-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             put-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              Lo-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             ca-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             tion-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             Mes-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             sage-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             Con-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             tent
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             In-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             putV-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             enueMes-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             sage-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             Con-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             tent
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             In-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             put-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             Con-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             tactMes-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             sage-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             Con-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             tent
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             In-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              putIn-
```

2.3. Bot API

30deMes-

sage-Con-

tent

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 <code>input\_message\_content</code> to send a message with the specified content instead of the sticker.

Source: https://core.telegram.org/bots/api#inlinequeryresultcachedsticker

```
type: Literal[InlineQueryResultType.STICKER]
```

Type of the result, must be sticker

id: str

Unique identifier for this result, 1-64 bytes

sticker\_file\_id: str

A valid file identifier of the sticker

## model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

 $model_post_init(context: Any, /) \rightarrow None$ 

We need to both initialize private attributes and call the user-defined model\_post\_init method.

reply\_markup: InlineKeyboardMarkup | None

Optional. Inline keyboard attached to the message

input\_message\_content: InputTextMessageContent | InputLocationMessageContent |
InputVenueMessageContent | InputContactMessageContent | InputInvoiceMessageContent |
None

Optional. Content of the message to be sent instead of the sticker

InlineQueryResultCachedVideo

```
class aiogram.types.inline_query_result_cached_video.InlineQueryResultCachedVideo(*, type:
                                                                                                     ~typ-
                                                                                                     ing.Literal[InlineQueryRest
                                                                                                     = Inline-
                                                                                                     QueryRe-
                                                                                                     sult-
                                                                                                     Type.VIDEO,
                                                                                                     id: str,
                                                                                                     video_file_id:
                                                                                                     str, title:
                                                                                                     str, de-
                                                                                                     scription:
                                                                                                     str | None
                                                                                                     = None,
                                                                                                     caption:
                                                                                                     str | None
                                                                                                     = None,
                                                                                                     parse_mode:
                                                                                                     ~aiogram.client.default.Def
                                                                                                     | None =
                                                                                                     <De-
                                                                                                     fault('parse_mode')>,
                                                                                                     cap-
                                                                                                     tion_entities:
                                                                                                     ~typ-
                                                                                                     ing.List[~aiogram.types.me.
                                                                                                     | None =
                                                                                                     None,
                                                                                                     show_caption_above_media
                                                                                                     ~aiogram.client.default.Def
                                                                                                     | None =
                                                                                                     <De-
                                                                                                     fault('show_caption_above_
                                                                                                     ply_markup:
                                                                                                     ~aiogram.types.inline_keyb
                                                                                                     | None =
                                                                                                     None, in-
                                                                                                     put_message_content:
                                                                                                     ~aiogram.types.input_text_i
                                                                                                     ~aiogram.types.input_locat
                                                                                                     ~aiogram.types.input_venue
                                                                                                     ~aiogram.types.input_conta
                                                                                                     ~aiogram.types.input_invoid
                                                                                                     | None =
                                                                                                     None,
                                                                                                     **ex-
                                                                                                     tra_data:
                                                                                                     ~typ-
                                                                                                     ing.Any)
```

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 <code>input\_message\_content</code> to send a message with the specified content instead of the video.

Source: https://core.telegram.org/bots/api#inlinequeryresultcachedvideo

#### type: Literal[InlineQueryResultType.VIDE0]

Type of the result, must be video

#### id: str

Unique identifier for this result, 1-64 bytes

#### video\_file\_id: str

A valid file identifier for the video file

#### title: str

Title for the result

## description: str | None

Optional. Short description of the result

#### caption: str | None

Optional. Caption of the video to be sent, 0-1024 characters after entities parsing

## model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

## parse\_mode: str | Default | None

Optional. Mode for parsing entities in the video caption. See formatting options for more details.

### caption\_entities: List[MessageEntity] | None

Optional. List of special entities that appear in the caption, which can be specified instead of parse\_mode

### show\_caption\_above\_media: bool | Default | None

Optional. Pass True, if the caption must be shown above the message media

# reply\_markup: InlineKeyboardMarkup | None

Optional. Inline keyboard attached to the message

```
input_message_content: InputTextMessageContent | InputLocationMessageContent |
InputVenueMessageContent | InputContactMessageContent | InputInvoiceMessageContent |
None
```

Optional. Content of the message to be sent instead of the video

# InlineQueryResultCachedVoice

```
\textbf{class} \  \, \textbf{aiogram.types.inline\_query\_result\_cached\_voice.} \\ \textbf{InlineQueryResultCachedVoice}(*, \textit{type:} \\ \textbf{aiogram.types.inline\_query\_result\_cached\_voice.} \\ \textbf{InlineQueryResultCachedVoice}(*, \textit{type:} \\ \textbf{aiogram.types.inline\_query\_result\_cached\_voice.} \\ \textbf{InlineQueryResultCachedVoice}(*, \textit{type:} \\ \textbf{aiogram.types.inline\_query\_result\_cached\_voice.} \\ \textbf{aiogram.types.type.} \\ \textbf{aiogram.types.type.} \\ \textbf{aiogram.types.type.} \\ \textbf{aiogram.type.type.} \\ \textbf{aiogram.type.} \\ \textbf{aiogram.type.type.} \\ \textbf{aiogram.type.} \\ \textbf{aiogram.type.type.} \\ \textbf{aiogram.type.} \\ \textbf{aiogram.type.type.} \\ \textbf{aiogram.type.} \\ \textbf{aiogram.type.type.} \\ \textbf{aiogram.type.} \\ \textbf{a
```

```
~typ-
ing.Literal[InlineQueryRest
= Inline-
QueryRe-
sult-
Type.VOICE,
id: str,
voice_file_id:
str, title:
str.
caption:
str | None
= None,
parse_mode:
str |
~aiogram.client.default.Def
|None|
<De-
fault('parse_mode')>,
cap-
tion\_entities:
~typ-
ing.List[~aiogram.types.me.
|None|
None, re-
ply_markup:
~aiogram.types.inline_keyb
|None|
None, in-
put_message_content:
~aiogram.types.input_text_t
~aiogram.types.input_locat
~aiogram.types.input_venue
~aiogram.types.input_conta
~aiogram.types.input_invoid
|None|
None,
**ex-
tra_data:
~typ-
ing.Any)
```

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 <code>input\_message\_content</code> to send a message with the specified content instead of the voice message.

Source: https://core.telegram.org/bots/api#inlinequeryresultcachedvoice

# type: Literal[InlineQueryResultType.VOICE]

Type of the result, must be *voice* 

```
id: str
     Unique identifier for this result, 1-64 bytes
voice_file_id: str
     A valid file identifier for the voice message
title: str
     Voice message title
caption: str | None
     Optional. Caption, 0-1024 characters after entities parsing
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
     A dictionary of computed field names and their corresponding ComputedFieldInfo objects.
model_post_init(context: Any, /) \rightarrow None
     We need to both initialize private attributes and call the user-defined model_post_init method.
parse_mode: str | Default | None
     Optional. Mode for parsing entities in the voice message caption. See formatting options for more details.
caption_entities: List[MessageEntity] | None
     Optional. List of special entities that appear in the caption, which can be specified instead of parse_mode
reply_markup: InlineKeyboardMarkup | None
     Optional. Inline keyboard attached to the message
input_message_content: InputTextMessageContent | InputLocationMessageContent |
InputVenueMessageContent | InputContactMessageContent | InputInvoiceMessageContent |
None
```

Optional. Content of the message to be sent instead of the voice message

### InlineQueryResultContact

class aiogram.types.inline\_query\_result\_contact.InlineQueryResultContact(\*, type: Lit-

eral[InlineQueryResultType.CONTACT = InlineQueryResult-Type.CONTACT, id: str, phone\_number: str, first\_name: str, last name: str | None = None, vcard: strNone = None. reply\_markup: InlineKeyboardMarkup | None = None, input\_message\_content: InputTextMessage-Content | InputLocationMessageContent | InputVenueMessage-Content | InputContactMessageContent | InputInvoiceMessageContent | None = None, thumbnail url:  $str \mid None = None,$ thumbnail width: int | None = None,thumbnail\_height:  $int \mid None = None,$ \*\*extra\_data: Any)

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.

Source: https://core.telegram.org/bots/api#inlinequeryresultcontact

## type: Literal[InlineQueryResultType.CONTACT]

Type of the result, must be *contact* 

id: str

Unique identifier for this result, 1-64 Bytes

phone\_number: str

Contact's phone number

first\_name: str

Contact's first name

last\_name: str | None

Optional. Contact's last name

vcard: str | None

Optional. Additional data about the contact in the form of a vCard, 0-2048 bytes

model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) → None

We need to both initialize private attributes and call the user-defined model_post_init method.

reply_markup: InlineKeyboardMarkup | None

Optional. Inline keyboard attached to the message

input_message_content: InputTextMessageContent | InputLocationMessageContent |
InputVenueMessageContent | InputContactMessageContent | InputInvoiceMessageContent |
None

Optional. Content of the message to be sent instead of the contact

thumbnail_url: str | None

Optional. Url of the thumbnail for the result

thumbnail_width: int | None

Optional. Thumbnail width

thumbnail_height: int | None

Optional. Thumbnail height
```

InlineQueryResultDocument

```
= InlineQueryRe-
sult-
Type.DOCUMENT,
id: str, title: str,
document_url: str,
mime_type: str,
caption: str | None
= None,
parse_mode: str |
~aiogram.client.default.Default
|None = < De-
fault('parse_mode')>,
caption_entities:
~typ-
ing.List[~aiogram.types.message_en
| None = None,
description: str |
None = None,
reply_markup:
~aiogram.types.inline_keyboard_max
| None = None, in-
put_message_content:
~aiogram.types.input_text_message_
~aiogram.types.input_location_mess
~aiogram.types.input_venue_message
~aiogram.types.input_contact_messa
~aiogram.types.input_invoice_messa
| None = None,
thumbnail url: str
| None = None,
thumbnail width:
int \mid None = None,
thumbnail_height:
int \mid None = None,
**extra data:
~typing.Any)
```

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

Source: https://core.telegram.org/bots/api#inlinequeryresultdocument

## type: Literal[InlineQueryResultType.DOCUMENT]

Type of the result, must be document

id: str

Unique identifier for this result, 1-64 bytes

title: str

```
Title for the result
     document url: str
          A valid URL for the file
     mime_type: str
          MIME type of the content of the file, either 'application/pdf' or 'application/zip'
     caption: str | None
          Optional. Caption of the document to be sent, 0-1024 characters after entities parsing
     parse_mode: str | Default | None
          Optional. Mode for parsing entities in the document caption. See formatting options for more details.
     model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
          A dictionary of computed field names and their corresponding ComputedFieldInfo objects.
     model_post_init(context: Any, /) \rightarrow None
          We need to both initialize private attributes and call the user-defined model post init method.
     caption_entities: List[MessageEntity] | None
          Optional. List of special entities that appear in the caption, which can be specified instead of parse_mode
     description: str | None
          Optional. Short description of the result
     reply_markup: InlineKeyboardMarkup | None
          Optional. Inline keyboard attached to the message
     input_message_content: InputTextMessageContent | InputLocationMessageContent |
     InputVenueMessageContent | InputContactMessageContent | InputInvoiceMessageContent |
     None
          Optional. Content of the message to be sent instead of the file
     thumbnail_url: str | None
          Optional. URL of the thumbnail (JPEG only) for the file
     thumbnail_width: int | None
          Optional. Thumbnail width
     thumbnail_height: int | None
          Optional. Thumbnail height
InlineQueryResultGame
class aiogram.types.inline_query_result_game.InlineQueryResultGame(*, type: Lit-
                                                                              eral[InlineQueryResultType.GAME]
                                                                              = InlineQueryResult-
                                                                              Type.GAME, id: str,
                                                                              game_short_name: str,
                                                                              reply markup:
                                                                              InlineKeyboardMarkup |
                                                                              None = None, **extra\_data:
```

Any)

# Represents a Game.

Source: https://core.telegram.org/bots/api#inlinequeryresultgame

## type: Literal[InlineQueryResultType.GAME]

Type of the result, must be game

## id: str

Unique identifier for this result, 1-64 bytes

# model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

# game\_short\_name: str

Short name of the game

# reply\_markup: InlineKeyboardMarkup | None

Optional. Inline keyboard attached to the message

# InlineQueryResultGif

```
class aiogram.types.inline_query_result_gif.InlineQueryResultGif(*, type: ~typ-
                                                                                  ing.Literal[InlineQueryResultType.GIF]
                                                                                  = InlineQueryResultType.GIF,
                                                                                  id: str, gif_url: str,
                                                                                  thumbnail_url: str, gif_width:
                                                                                  int | None = None, gif_height:
                                                                                  int \mid None = None,
                                                                                  gif\_duration: int \mid None =
                                                                                  None, thumbnail_mime_type:
                                                                                  str \mid None = None, title: str \mid
                                                                                  None = None, caption: str \mid
                                                                                  None = None, parse\_mode: str
                                                                                  ~aiogram.client.default.Default
                                                                                  | None =
                                                                                  <Default('parse_mode')>,
                                                                                  caption_entities: ~typ-
                                                                                  ing.List[~aiogram.types.message_entity.MessageE
                                                                                  | None = None,
                                                                                  show_caption_above_media:
                                                                                  bool |
                                                                                  ~aiogram.client.default.Default
                                                                                  |None = < De-
                                                                                  fault('show caption above media')>,
                                                                                  reply_markup:
                                                                                  ~aiogram.types.inline_keyboard_markup.InlineKe
                                                                                  | None = None,
                                                                                  input_message_content:
                                                                                  ~aiogram.types.input_text_message_content.Input
                                                                                  ~aiogram.types.input_location_message_content.l
                                                                                  ~aiogram.types.input_venue_message_content.Inp
                                                                                  ~aiogram.types.input_contact_message_content.In
                                                                                  ~aiogram.types.input_invoice_message_content.Ir
                                                                                  | None = None, **extra_data:
                                                                                  ~typing.Any)
```

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 <code>input\_message\_content</code> to send a message with the specified content instead of the animation.

Source: https://core.telegram.org/bots/api#inlinequeryresultgif

```
type: Literal[InlineQueryResultType.GIF]
```

Type of the result, must be gif

id: str

Unique identifier for this result, 1-64 bytes

gif\_url: str

A valid URL for the GIF file. File size must not exceed 1MB

thumbnail\_url: str

```
URL of the static (JPEG or GIF) or animated (MPEG4) thumbnail for the result
gif_width: int | None
     Optional. Width of the GIF
gif_height: int | None
     Optional. Height of the GIF
gif_duration: int | None
     Optional. Duration of the GIF in seconds
thumbnail_mime_type: str | None
     Optional. MIME type of the thumbnail, must be one of 'image/jpeg', 'image/gif', or 'video/mp4'. Defaults
     to 'image/jpeg'
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
     A dictionary of computed field names and their corresponding ComputedFieldInfo objects.
model_post_init(context: Any, /) \rightarrow None
     We need to both initialize private attributes and call the user-defined model_post_init method.
title: str | None
     Optional. Title for the result
caption: str | None
     Optional. Caption of the GIF file to be sent, 0-1024 characters after entities parsing
parse_mode: str | Default | None
     Optional. Mode for parsing entities in the caption. See formatting options for more details.
caption_entities: List[MessageEntity] | None
     Optional. List of special entities that appear in the caption, which can be specified instead of parse_mode
show_caption_above_media: bool | Default | None
     Optional. Pass True, if the caption must be shown above the message media
reply_markup: InlineKeyboardMarkup | None
     Optional. Inline keyboard attached to the message
input_message_content: InputTextMessageContent | InputLocationMessageContent |
InputVenueMessageContent | InputContactMessageContent | InputInvoiceMessageContent |
None
     Optional. Content of the message to be sent instead of the GIF animation
```

Optional. Content of the message to be sent instead of the Gir animation

## **InlineQueryResultLocation**

class aiogram.types.inline\_query\_result\_location.InlineQueryResultLocation(\*, type: Lit-

eral[InlineQueryResultType.LOCATI = InlineQueryResult-Type.LOCATION, id: str, latitude: float, longitude: float, title: str, horizontal\_accuracy: float | None = None,live\_period: int | None = None, heading: int | None = None, proximity\_alert\_radius:  $int \mid None = None,$ reply\_markup: InlineKeyboard-Markup | None = None, input\_message\_content: InputTextMessageContent | InputLocation-MessageContent | InputVenueMessageContent | InputContactMessageContent | InputInvoiceMessageContent | None = None,thumbnail url: str | None = None,thumbnail width:  $int \mid None = None,$ thumbnail\_height:

int | None = None,
\*\*extra data:

Any)

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.

Source: https://core.telegram.org/bots/api#inlinequeryresultlocation

# type: Literal[InlineQueryResultType.LOCATION]

Type of the result, must be location

## id: str

Unique identifier for this result, 1-64 Bytes

# latitude: float

Location latitude in degrees

longitude: float

Location longitude in degrees

title: str

Location title

horizontal\_accuracy: float | None

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

live\_period: int | None

*Optional*. Period in seconds during which the location can be updated, should be between 60 and 86400, or 0x7FFFFFFF for live locations that can be edited indefinitely.

model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

 $model_post_init(context: Any, /) \rightarrow None$ 

We need to both initialize private attributes and call the user-defined model\_post\_init method.

heading: int | None

*Optional*. For live locations, a direction in which the user is moving, in degrees. Must be between 1 and 360 if specified.

proximity\_alert\_radius: int | None

*Optional*. For live locations, a maximum distance for proximity alerts about approaching another chat member, in meters. Must be between 1 and 100000 if specified.

reply\_markup: InlineKeyboardMarkup | None

Optional. Inline keyboard attached to the message

input\_message\_content: InputTextMessageContent | InputLocationMessageContent |
InputVenueMessageContent | InputContactMessageContent | InputInvoiceMessageContent |
None

Optional. Content of the message to be sent instead of the location

thumbnail\_url: str | None

Optional. Url of the thumbnail for the result

thumbnail\_width: int | None *Optional*. Thumbnail width

thumbnail\_height: int | None
 Optional. Thumbnail height

InlineQueryResultMpeg4Gif

```
class aiogram.types.inline_query_result_mpeg4_gif.InlineQueryResultMpeg4Gif(*, type: ~typ-
                                                                                             ing.Literal[InlineQueryResultType.I
                                                                                             = Inline-
                                                                                             QueryResult-
                                                                                             Type.MPEG4_GIF,
                                                                                             id: str,
                                                                                             mpeg4_url: str,
                                                                                             thumbnail_url:
                                                                                             str,
                                                                                             mpeg4_width:
                                                                                             int \mid None =
                                                                                             None,
                                                                                             mpeg4_height:
                                                                                             int | None =
                                                                                             None,
                                                                                             mpeg4_duration:
                                                                                             int | None =
                                                                                             None, thumb-
                                                                                             nail_mime_type:
                                                                                             str \mid None =
                                                                                             None, title: str |
                                                                                             None = None,
                                                                                             caption: str |
                                                                                             None = None,
                                                                                             parse_mode: str |
                                                                                             ~aiogram.client.default.Default
                                                                                             |None = < De-
                                                                                             fault('parse_mode')>,
                                                                                             caption_entities:
                                                                                             ~typ-
                                                                                             ing.List[~aiogram.types.message_e.
                                                                                             | None = None,
                                                                                             show_caption_above_media:
                                                                                             bool |
                                                                                             ~aiogram.client.default.Default
                                                                                             |None = < De-
                                                                                             fault('show_caption_above_media')
                                                                                             reply_markup:
                                                                                             ~aiogram.types.inline_keyboard_m
                                                                                             | None = None,
                                                                                             in-
                                                                                             put_message_content:
                                                                                             ~aiogram.types.input_text_message
                                                                                             ~aiogram.types.input_location_mes
                                                                                             ~aiogram.types.input_venue_messa
                                                                                             ~aiogram.types.input_contact_mess
                                                                                             ~aiogram.types.input_invoice_mess
                                                                                             | None = None,
                                                                                              **extra_data:
```

~typing.Any)

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 <code>input\_message\_content</code> to send a message with the specified content instead of the animation.

Source: https://core.telegram.org/bots/api#inlinequeryresultmpeg4gif

# type: Literal[InlineQueryResultType.MPEG4\_GIF]

Type of the result, must be mpeg4\_gif

#### id: str

Unique identifier for this result, 1-64 bytes

### mpeg4\_url: str

A valid URL for the MPEG4 file. File size must not exceed 1MB

## thumbnail\_url: str

URL of the static (JPEG or GIF) or animated (MPEG4) thumbnail for the result

# mpeg4\_width: int | None

Optional. Video width

### mpeg4\_height: int | None

Optional. Video height

# mpeg4\_duration: int | None

Optional. Video duration in seconds

# thumbnail\_mime\_type: str | None

*Optional*. MIME type of the thumbnail, must be one of 'image/jpeg', 'image/gif', or 'video/mp4'. Defaults to 'image/jpeg'

## model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

### $model_post_init(context: Any, /) \rightarrow None$

We need to both initialize private attributes and call the user-defined model\_post\_init method.

## title: str | None

Optional. Title for the result

### caption: str | None

Optional. Caption of the MPEG-4 file to be sent, 0-1024 characters after entities parsing

## parse\_mode: str | Default | None

Optional. Mode for parsing entities in the caption. See formatting options for more details.

## caption\_entities: List[MessageEntity] | None

Optional. List of special entities that appear in the caption, which can be specified instead of parse mode

## show\_caption\_above\_media: bool | Default | None

Optional. Pass True, if the caption must be shown above the message media

## reply\_markup: InlineKeyboardMarkup | None

Optional. Inline keyboard attached to the message

input\_message\_content: InputTextMessageContent | InputLocationMessageContent |
InputVenueMessageContent | InputContactMessageContent | InputInvoiceMessageContent |
None

Optional. Content of the message to be sent instead of the video animation

# InlineQueryResultPhoto

```
class aiogram.types.inline_query_result_photo.InlineQueryResultPhoto(*, type: ~type-
                                                                                     ing.Literal[InlineQueryResultType.PHOTO]
                                                                                     = Inline Query Result-
                                                                                     Type.PHOTO, id: str,
                                                                                     photo_url: str,
                                                                                     thumbnail_url: str,
                                                                                     photo_width: int | None =
                                                                                     None, photo_height: int |
                                                                                     None = None, title: str
                                                                                     None = None, description:
                                                                                     str \mid None = None,
                                                                                     caption: str | None =
                                                                                     None, parse mode: str |
                                                                                     ~aiogram.client.default.Default
                                                                                     |None|
                                                                                     <Default('parse_mode')>,
                                                                                     caption_entities: ~typ-
                                                                                     ing.List[~aiogram.types.message_entity.Mes.
                                                                                     | None = None,
                                                                                     show_caption_above_media:
                                                                                     bool |
                                                                                     ~aiogram.client.default.Default
                                                                                     |None = < De-
                                                                                    fault('show_caption_above_media')>,
                                                                                     reply_markup:
                                                                                     ~aiogram.types.inline_keyboard_markup.Inl
                                                                                     | None = None,
                                                                                     input_message_content:
                                                                                     ~aiogram.types.input_text_message_content.
                                                                                     ~aiogram.types.input_location_message_com
                                                                                     ~aiogram.types.input_venue_message_conten
                                                                                     ~aiogram.types.input_contact_message_cont
                                                                                     ~aiogram.types.input_invoice_message_cont
                                                                                     | None = None,
                                                                                     **extra_data:
                                                                                     ~typing.Any)
```

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.

Source: https://core.telegram.org/bots/api#inlinequeryresultphoto

```
type: Literal[InlineQueryResultType.PHOTO]
```

Type of the result, must be photo

#### id: str

Unique identifier for this result, 1-64 bytes

#### photo\_url: str

A valid URL of the photo. Photo must be in JPEG format. Photo size must not exceed 5MB

```
thumbnail_url: str
     URL of the thumbnail for the photo
photo_width: int | None
     Optional. Width of the photo
photo_height: int | None
     Optional. Height of the photo
title: str | None
     Optional. Title for the result
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
     A dictionary of computed field names and their corresponding ComputedFieldInfo objects.
model_post_init(context: Any, /) \rightarrow None
     We need to both initialize private attributes and call the user-defined model_post_init method.
description: str | None
     Optional. Short description of the result
caption: str | None
     Optional. Caption of the photo to be sent, 0-1024 characters after entities parsing
parse_mode: str | Default | None
     Optional. Mode for parsing entities in the photo caption. See formatting options for more details.
caption_entities: List[MessageEntity] | None
     Optional. List of special entities that appear in the caption, which can be specified instead of parse_mode
show_caption_above_media: bool | Default | None
     Optional. Pass True, if the caption must be shown above the message media
reply_markup: InlineKeyboardMarkup | None
     Optional. Inline keyboard attached to the message
input_message_content: InputTextMessageContent | InputLocationMessageContent |
InputVenueMessageContent | InputContactMessageContent | InputInvoiceMessageContent |
None
```

Optional. Content of the message to be sent instead of the photo

# **InlineQueryResultVenue**

class aiogram.types.inline\_query\_result\_venue.InlineQueryResultVenue(\*, type: Lit-

eral[InlineQueryResultType.VENUE]

= In line Query Result-

Type.VENUE, id: str,

*latitude: float, longitude:* 

float, title: str, address:

 $str, foursquare\_id: str \mid$ 

None = None,

foursquare\_type: str |

None = None,

google\_place\_id: str |

None = None,

google\_place\_type: str |

None = None,

reply\_markup:

InlineKeyboardMarkup |

None = None,

 $input\_message\_content:$ 

Input Text Message Content

| InputLocationMessage-

Content

InputVenueMessageCon-

tent

InputContactMessageCon-

tent

InputInvoiceMessageCon-

 $tent \mid None = None,$ 

thumbnail\_url: str | None

 $= None, thumbnail\_width:$ 

 $int \mid None = None,$ 

thumbnail\_height: int |

None = None,

\*\*extra\_data: Any)

Represents a venue. By default, the venue will be sent by the user. Alternatively, you can use *in-put\_message\_content* to send a message with the specified content instead of the venue.

Source: https://core.telegram.org/bots/api#inlinequeryresultvenue

# type: Literal[InlineQueryResultType.VENUE]

Type of the result, must be venue

id: str

Unique identifier for this result, 1-64 Bytes

latitude: float

Latitude of the venue location in degrees

longitude: float

Longitude of the venue location in degrees

title: str

Title of the venue

address: str

Address of the venue

```
foursquare_id: str | None
    Optional. Foursquare identifier of the venue if known
foursquare_type: str | None
               Foursquare type of the venue, if known.
                                                          (For example, 'arts_entertainment/default',
    'arts entertainment/aquarium' or 'food/icecream'.)
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
    A dictionary of computed field names and their corresponding ComputedFieldInfo objects.
model_post_init(context: Any, /) \rightarrow None
    We need to both initialize private attributes and call the user-defined model_post_init method.
google_place_id: str | None
    Optional. Google Places identifier of the venue
google_place_type: str | None
    Optional. Google Places type of the venue. (See supported types.)
reply_markup: InlineKeyboardMarkup | None
    Optional. Inline keyboard attached to the message
input_message_content: InputTextMessageContent | InputLocationMessageContent |
InputVenueMessageContent | InputContactMessageContent | InputInvoiceMessageContent |
None
    Optional. Content of the message to be sent instead of the venue
thumbnail_url: str | None
    Optional. Url of the thumbnail for the result
thumbnail_width: int | None
    Optional. Thumbnail width
thumbnail_height: int | None
    Optional. Thumbnail height
```

# InlineQueryResultVideo

```
class aiogram.types.inline_query_result_video.InlineQueryResultVideo(*, type: ~type-
                                                                                      ing.Literal[InlineQueryResultType.VIDEO]
                                                                                      = Inline Query Result-
                                                                                      Type.VIDEO, id: str,
                                                                                      video_url: str, mime_type:
                                                                                      str, thumbnail_url: str,
                                                                                      title: str, caption: str |
                                                                                      None = None,
                                                                                      parse_mode: str |
                                                                                      ~aiogram.client.default.Default
                                                                                      | None =
                                                                                      <Default('parse_mode')>,
                                                                                      caption entities: ~typ-
                                                                                      ing.List[~aiogram.types.message_entity.Mes.
                                                                                      | None = None,
                                                                                      show_caption_above_media:
                                                                                      bool |
                                                                                      ~aiogram.client.default.Default
                                                                                      |None = < De-
                                                                                     fault('show_caption_above_media')>,
                                                                                      video\_width: int \mid None =
                                                                                      None, video_height: int |
                                                                                      None = None,
                                                                                      video_duration: int | None
                                                                                      = None, description: str |
                                                                                      None = None,
                                                                                      reply_markup:
                                                                                      ~aiogram.types.inline_keyboard_markup.Inl
                                                                                      | None = None,
                                                                                      input_message_content:
                                                                                      ~aiogram.types.input_text_message_content.
                                                                                      ~aiogram.types.input_location_message_com
                                                                                      ~aiogram.types.input_venue_message_conten
                                                                                      ~aiogram.types.input_contact_message_cont
                                                                                      ~aiogram.types.input_invoice_message_cont
                                                                                      | None = None,
```

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 <code>input\_message\_content</code> 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 <code>input\_message\_content</code>.

Source: https://core.telegram.org/bots/api#inlinequeryresultvideo

### type: Literal[InlineQueryResultType.VIDE0]

Type of the result, must be video

\*\*extra\_data: ~typing.Any)

```
id: str
     Unique identifier for this result, 1-64 bytes
video_url: str
     A valid URL for the embedded video player or video file
mime_type: str
    MIME type of the content of the video URL, 'text/html' or 'video/mp4'
thumbnail_url: str
     URL of the thumbnail (JPEG only) for the video
title: str
     Title for the result
caption: str | None
     Optional. Caption of the video to be sent, 0-1024 characters after entities parsing
parse_mode: str | Default | None
     Optional. Mode for parsing entities in the video caption. See formatting options for more details.
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
     A dictionary of computed field names and their corresponding ComputedFieldInfo objects.
model_post_init(context: Any, /) \rightarrow None
     We need to both initialize private attributes and call the user-defined model_post_init method.
caption_entities: List[MessageEntity] | None
     Optional. List of special entities that appear in the caption, which can be specified instead of parse_mode
show_caption_above_media: bool | Default | None
     Optional. Pass True, if the caption must be shown above the message media
video_width: int | None
     Optional. Video width
video_height: int | None
     Optional. Video height
video_duration: int | None
     Optional. Video duration in seconds
description: str | None
     Optional. Short description of the result
reply_markup: InlineKeyboardMarkup | None
     Optional. Inline keyboard attached to the message
input_message_content: InputTextMessageContent | InputLocationMessageContent |
InputVenueMessageContent | InputContactMessageContent | InputInvoiceMessageContent |
None
```

2.3. Bot API 323

sultVideo is used to send an HTML-page as a result (e.g., a YouTube video).

Optional. Content of the message to be sent instead of the video. This field is **required** if InlineQueryRe-

# **InlineQueryResultVoice**

```
class aiogram.types.inline_query_result_voice.InlineQueryResultVoice(*, type: ~type-
                                                                                     ing.Literal[InlineQueryResultType.VOICE]
                                                                                     = Inline Query Result-
                                                                                     Type.VOICE, id: str,
                                                                                     voice_url: str, title: str,
                                                                                     caption: str | None =
                                                                                     None, parse_mode: str |
                                                                                     ~aiogram.client.default.Default
                                                                                     | None =
                                                                                     <Default('parse_mode')>,
                                                                                     caption_entities: ~typ-
                                                                                     ing.List[~aiogram.types.message_entity.Mes.
                                                                                     | None = None,
                                                                                     voice_duration: int | None
                                                                                     = None, reply_markup:
                                                                                     ~aiogram.types.inline_keyboard_markup.Inl
                                                                                     | None = None,
                                                                                     input_message_content:
                                                                                     ~aiogram.types.input_text_message_content.
                                                                                     ~aiogram.types.input_location_message_com
                                                                                     ~aiogram.types.input_venue_message_conten
                                                                                     ~aiogram.types.input_contact_message_cont
                                                                                     ~aiogram.types.input_invoice_message_cont
                                                                                     | None = None,
                                                                                      **extra data:
                                                                                     ~typing.Any)
     Represents a link to a voice recording in an .OGG container encoded with OPUS. By default, this voice recording
```

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 <code>input\_message\_content</code> to send a message with the specified content instead of the the voice message.

Source: https://core.telegram.org/bots/api#inlinequeryresultvoice

```
type: Literal[InlineQueryResultType.VOICE]
```

Type of the result, must be voice

id: str

Unique identifier for this result, 1-64 bytes

voice\_url: str

A valid URL for the voice recording

title: str

Recording title

caption: str | None

Optional. Caption, 0-1024 characters after entities parsing

model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
parse_mode: str | Default | None
```

Optional. Mode for parsing entities in the voice message caption. See formatting options for more details.

```
caption_entities: List[MessageEntity] | None
```

Optional. List of special entities that appear in the caption, which can be specified instead of parse\_mode

```
voice_duration: int | None
```

Optional. Recording duration in seconds

```
reply_markup: InlineKeyboardMarkup | None
```

Optional. Inline keyboard attached to the message

```
input_message_content: InputTextMessageContent | InputLocationMessageContent |
InputVenueMessageContent | InputContactMessageContent | InputInvoiceMessageContent |
None
```

Optional. Content of the message to be sent instead of the voice recording

## **InlineQueryResultsButton**

 $\textbf{class} \ \ aiogram. types. in line\_query\_results\_button. \textbf{InlineQueryResultsButton} (*, \textit{text: str}, \textit{web\_app: } line\_query\_results\_button) (*, \textit{text: str}, \textit{text: str$ 

```
WebAppInfo | None
```

= None,

start\_parameter: str |
None = None,

\*\*extra data: Any)

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

Source: https://core.telegram.org/bots/api#inlinequeryresultsbutton

## text: str

Label text on the button

```
web_app: WebAppInfo | None
```

*Optional*. Description of the Web App that will be launched when the user presses the button. The Web App will be able to switch back to the inline mode using the method switchInlineQuery inside the Web App.

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
start_parameter: str | None
```

*Optional*. Deep-linking parameter for the /start message sent to the bot when a user presses the button. 1-64 characters, only A-Z, a-z, 0-9, \_ and - are allowed.

# InputContactMessageContent

```
class aiogram.types.input_contact_message_content.InputContactMessageContent(*,
```

phone\_number: str, first\_name: str, last\_name: str | None = None, vcard: str | None = None, \*\*extra\_data: Any)

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

Source: https://core.telegram.org/bots/api#inputcontactmessagecontent

phone\_number: str

Contact's phone number

first\_name: str

Contact's first name

model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

 $model_post_init(context: Any, /) \rightarrow None$ 

We need to both initialize private attributes and call the user-defined model\_post\_init method.

last\_name: str | None

Optional. Contact's last name

vcard: str | None

Optional. Additional data about the contact in the form of a vCard, 0-2048 bytes

### InputInvoiceMessageContent

is\_flexible: bool | None = None,\*\*extra\_data:

Any)

```
class aiogram.types.input_invoice_message_content.InputInvoiceMessageContent(*, title: str,
                                                                                              description: str,
                                                                                              payload: str,
                                                                                              currency: str,
                                                                                              prices:
                                                                                              List[LabeledPrice],
                                                                                              provider_token:
                                                                                              str | None =
                                                                                              None,
                                                                                              max_tip_amount:
                                                                                              int \mid None =
                                                                                              None, sug-
                                                                                              gested_tip_amounts:
                                                                                              List[int] | None
                                                                                              = None,
                                                                                              provider_data:
                                                                                              str | None =
                                                                                              None,
                                                                                              photo_url: str |
                                                                                              None = None,
                                                                                              photo_size: int |
                                                                                              None = None,
                                                                                              photo_width:
                                                                                              int | None =
                                                                                              None,
                                                                                              photo_height:
                                                                                              int \mid None =
                                                                                              None,
                                                                                              need_name:
                                                                                              bool | None =
                                                                                              None,
                                                                                              need_phone_number:
                                                                                              bool | None =
                                                                                              None,
                                                                                              need email:
                                                                                              bool \mid None =
                                                                                              None,
                                                                                              need_shipping_address:
                                                                                              bool \mid None =
                                                                                              None,
                                                                                              send_phone_number_to_provider:
                                                                                              bool | None =
                                                                                              None.
                                                                                              send_email_to_provider:
                                                                                              bool | None =
                                                                                              None,
```

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

Source: https://core.telegram.org/bots/api#inputinvoicemessagecontent

title: str

Product name, 1-32 characters

### description: str

Product description, 1-255 characters

### payload: str

Bot-defined invoice payload, 1-128 bytes. This will not be displayed to the user, use it for your internal processes.

### currency: str

Three-letter ISO 4217 currency code, see more on currencies. Pass 'XTR' for payments in Telegram Stars.

## prices: List[LabeledPrice]

Price breakdown, a JSON-serialized list of components (e.g. product price, tax, discount, delivery cost, delivery tax, bonus, etc.). Must contain exactly one item for payments in Telegram Stars.

# provider\_token: str | None

Optional. Payment provider token, obtained via @BotFather. Pass an empty string for payments in Telegram Stars.

#### max\_tip\_amount: int | None

Optional. The maximum accepted amount for tips in the *smallest units* of the currency (integer, **not** float/double). For example, for a maximum tip of US\$ 1.45 pass max\_tip\_amount = 145. See the *exp* parameter in currencies.json, it shows the number of digits past the decimal point for each currency (2 for the majority of currencies). Defaults to 0. Not supported for payments in Telegram Stars.

### suggested\_tip\_amounts: List[int] | None

*Optional*. A JSON-serialized array of suggested amounts of tip in the *smallest units* of the currency (integer, **not** float/double). At most 4 suggested tip amounts can be specified. The suggested tip amounts must be positive, passed in a strictly increased order and must not exceed *max\_tip\_amount*.

### provider\_data: str | None

*Optional.* A JSON-serialized object for data about the invoice, which will be shared with the payment provider. A detailed description of the required fields should be provided by the payment provider.

## photo\_url: str | None

*Optional*. URL of the product photo for the invoice. Can be a photo of the goods or a marketing image for a service.

#### model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

# $model_post_init(context: Any, /) \rightarrow None$

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### photo\_size: int | None

Optional. Photo size in bytes

# photo\_width: int | None

Optional. Photo width

### photo\_height: int | None

Optional. Photo height

# need\_name: bool | None

*Optional.* Pass True if you require the user's full name to complete the order. Ignored for payments in Telegram Stars.

#### need\_phone\_number: bool | None

*Optional.* Pass True if you require the user's phone number to complete the order. Ignored for payments in Telegram Stars.

### need\_email: bool | None

*Optional*. Pass True if you require the user's email address to complete the order. Ignored for payments in Telegram Stars.

# need\_shipping\_address: bool | None

*Optional*. Pass True if you require the user's shipping address to complete the order. Ignored for payments in Telegram Stars.

# send\_phone\_number\_to\_provider: bool | None

*Optional*. Pass True if the user's phone number should be sent to the provider. Ignored for payments in Telegram Stars.

## send\_email\_to\_provider: bool | None

*Optional.* Pass True if the user's email address should be sent to the provider. Ignored for payments in Telegram Stars.

# is\_flexible: bool | None

*Optional*. Pass True if the final price depends on the shipping method. Ignored for payments in Telegram Stars.

# InputLocationMessageContent

class aiogram.types.input\_location\_message\_content.InputLocationMessageContent(\*, latitude:

longitude: float, horizontal\_accuracy: float | None = None.live period: *int* | *None* = None, heading: int | None = None, proximity\_alert\_radius: *int* | *None* = None, \*\*extra\_data:

Any)

float,

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

Source: https://core.telegram.org/bots/api#inputlocationmessagecontent

latitude: float

Latitude of the location in degrees

longitude: float

Longitude of the location in degrees

#### horizontal\_accuracy: float | None

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

# model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### live\_period: int | None

*Optional.* Period in seconds during which the location can be updated, should be between 60 and 86400, or 0x7FFFFFF for live locations that can be edited indefinitely.

### heading: int | None

*Optional.* For live locations, a direction in which the user is moving, in degrees. Must be between 1 and 360 if specified.

### proximity\_alert\_radius: int | None

*Optional*. For live locations, a maximum distance for proximity alerts about approaching another chat member, in meters. Must be between 1 and 100000 if specified.

## InputMessageContent

# class aiogram.types.input\_message\_content.InputMessageContent(\*\*extra\_data: Any)

This object represents the content of a message to be sent as a result of an inline query. Telegram clients currently support the following 5 types:

- aiogram.types.input\_text\_message\_content.InputTextMessageContent
- aiogram.types.input\_location\_message\_content.InputLocationMessageContent
- $\bullet \ aiogram.types.input\_venue\_message\_content.InputVenueMessageContent\\$
- aiogram.types.input\_contact\_message\_content.InputContactMessageContent
- aiogram.types.input\_invoice\_message\_content.InputInvoiceMessageContent

Source: https://core.telegram.org/bots/api#inputmessagecontent

# model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model post init method.

#### InputTextMessageContent

```
class aiogram.types.input_text_message_content.InputTextMessageContent(*, message_text: str,
                                                                                      parse_mode: str |
                                                                                       ~aiogram.client.default.Default
                                                                                       |None = < De-
                                                                                      fault('parse_mode')>,
                                                                                       entities: ~typ-
                                                                                       ing.List[~aiogram.types.message_entity.M
                                                                                       | None = None,
                                                                                       link_preview_options:
                                                                                       ~aiogram.types.link_preview_options.Link
                                                                                       ~aiogram.client.default.Default
                                                                                       |None = < De-
                                                                                      fault('link_preview')>,
                                                                                       dis-
                                                                                       able_web_page_preview:
                                                                                       bool \mid None = None,
                                                                                       **extra data:
                                                                                       ~typing.Any)
     Represents the content of a text message to be sent as the result of an inline query.
     Source: https://core.telegram.org/bots/api#inputtextmessagecontent
     message_text: str
          Text of the message to be sent, 1-4096 characters
     parse_mode: str | Default | None
           Optional. Mode for parsing entities in the message text. See formatting options for more details.
     entities: List[MessageEntity] | None
           Optional. List of special entities that appear in message text, which can be specified instead of parse_mode
     model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
           A dictionary of computed field names and their corresponding ComputedFieldInfo objects.
     model_post_init(context: Any, /) \rightarrow None
           We need to both initialize private attributes and call the user-defined model_post_init method.
     link_preview_options: LinkPreviewOptions | Default | None
           Optional. Link preview generation options for the message
     disable_web_page_preview: bool | None
           Optional. Disables link previews for links in the sent message
           Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023
```

InputVenueMessageContent

class aiogram.types.input\_venue\_message\_content.InputVenueMessageContent(\*, latitude: float,

longitude: float, title: str, address: str, foursquare\_id: str | None = None, foursquare\_type: str | None = None, google\_place\_id: str | None = None, google\_place\_type: str | None = None, \*\*extra\_data: Any)

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

Source: https://core.telegram.org/bots/api#inputvenuemessagecontent

latitude: float

Latitude of the venue in degrees

longitude: float

Longitude of the venue in degrees

title: str

Name of the venue

address: str

Address of the venue

model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

 $model_post_init(context: Any, /) \rightarrow None$ 

We need to both initialize private attributes and call the user-defined model\_post\_init method.

foursquare\_id: str | None

Optional. Foursquare identifier of the venue, if known

foursquare\_type: str | None

*Optional*. Foursquare type of the venue, if known. (For example, 'arts\_entertainment/default', 'arts\_entertainment/aquarium' or 'food/icecream'.)

google\_place\_id: str | None

Optional. Google Places identifier of the venue

google\_place\_type: str | None

Optional. Google Places type of the venue. (See supported types.)

### **SentWebAppMessage**

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

Source: https://core.telegram.org/bots/api#sentwebappmessage

```
inline_message_id: str | None
```

*Optional*. Identifier of the sent inline message. Available only if there is an inline keyboard attached to the message.

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### **Stickers**

## InputSticker

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

Source: https://core.telegram.org/bots/api#inputsticker

```
sticker: InputFile | str
```

The added sticker. Pass a *file\_id* as a String to send a file that already exists on the Telegram servers, pass an HTTP URL as a String for Telegram to get a file from the Internet, upload a new one using multipart/form-data, or pass 'attach://<file\_attach\_name>' to upload a new one using multipart/form-data under <file\_attach\_name> name. Animated and video stickers can't be uploaded via HTTP URL. *More information on Sending Files* »

```
format: str
```

Format of the added sticker, must be one of 'static' for a .WEBP or .PNG image, 'animated' for a .TGS animation, 'video' for a WEBM video

```
emoji_list: List[str]
```

List of 1-20 emoji associated with the sticker

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
mask_position: MaskPosition | None
```

Optional. Position where the mask should be placed on faces. For 'mask' stickers only.

```
keywords: List[str] | None
```

*Optional*. List of 0-20 search keywords for the sticker with total length of up to 64 characters. For 'regular' and 'custom\_emoji' stickers only.

### **MaskPosition**

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

Source: https://core.telegram.org/bots/api#maskposition

### point: str

The part of the face relative to which the mask should be placed. One of 'forehead', 'eyes', 'mouth', or 'chin'.

#### x\_shift: float

Shift by X-axis measured in widths of the mask scaled to the face size, from left to right. For example, choosing -1.0 will place mask just to the left of the default mask position.

### model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

# y\_shift: float

Shift by Y-axis measured in heights of the mask scaled to the face size, from top to bottom. For example, 1.0 will place the mask just below the default mask position.

#### scale: float

Mask scaling coefficient. For example, 2.0 means double size.

# Sticker

This object represents a sticker.

Source: https://core.telegram.org/bots/api#sticker

### file\_id: str

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

### file\_unique\_id: str

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

#### type: str

Type of the sticker, currently one of 'regular', 'mask', 'custom\_emoji'. The type of the sticker is independent from its format, which is determined by the fields *is\_animated* and *is\_video*.

#### width: int

Sticker width

```
height: int
     Sticker height
is_animated: bool
     True, if the sticker is animated
is_video: bool
     True, if the sticker is a video sticker
thumbnail: PhotoSize | None
     Optional. Sticker thumbnail in the .WEBP or .JPG format
emoji: str | None
     Optional. Emoji associated with the sticker
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
     A dictionary of computed field names and their corresponding ComputedFieldInfo objects.
model_post_init(context: Any, /) \rightarrow None
     We need to both initialize private attributes and call the user-defined model_post_init method.
set_name: str | None
     Optional. Name of the sticker set to which the sticker belongs
premium_animation: File | None
     Optional. For premium regular stickers, premium animation for the sticker
mask_position: MaskPosition | None
     Optional. For mask stickers, the position where the mask should be placed
custom_emoji_id: str | None
     Optional. For custom emoji stickers, unique identifier of the custom emoji
needs_repainting: bool | None
     Optional. True, if the sticker must be repainted to a text color in messages, the color of the Telegram
     Premium badge in emoji status, white color on chat photos, or another appropriate color in other places
file_size: int | None
     Optional. File size in bytes
set_position_in_set(position: int, **kwargs: Any) \rightarrow SetStickerPositionInSet
                                                 aiogram.methods.set_sticker_position_in_set.
     Shortcut
     SetStickerPositionInSet will automatically fill method attributes:

    sticker

     Use this method to move a sticker in a set created by the bot to a specific position. Returns True on success.
     Source: https://core.telegram.org/bots/api#setstickerpositioninset
         Parameters
             position – New sticker position in the set, zero-based
             instance
                          of
                                 method
                                             aiogram.methods.set_sticker_position_in_set.
             SetStickerPositionInSet
```

```
delete\_from\_set(**kwargs: Any) \rightarrow DeleteStickerFromSet
```

Shortcut for method aiogram.methods.delete\_sticker\_from\_set.DeleteStickerFromSet will automatically fill method attributes:

sticker

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

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

#### Returns

instance of method aiogram.methods.delete\_sticker\_from\_set.
DeleteStickerFromSet

#### **StickerSet**

This object represents a sticker set.

Source: https://core.telegram.org/bots/api#stickerset

name: str

Sticker set name

title: str

Sticker set title

sticker\_type: str

Type of stickers in the set, currently one of 'regular', 'mask', 'custom\_emoji'

stickers: List[Sticker]

List of all set stickers

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
thumbnail: PhotoSize | None
```

Optional. Sticker set thumbnail in the .WEBP, .TGS, or .WEBM format

is\_animated: bool | None

True, if the sticker set contains animated stickers

Deprecated since version API:7.2: https://core.telegram.org/bots/api-changelog#march-31-2024

is\_video: bool | None

True, if the sticker set contains video stickers

Deprecated since version API:7.2: https://core.telegram.org/bots/api-changelog#march-31-2024

### **Payments**

#### **Invoice**

This object contains basic information about an invoice.

Source: https://core.telegram.org/bots/api#invoice

title: str

Product name

description: str

Product description

start\_parameter: str

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

model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

 $model_post_init(context: Any, /) \rightarrow None$ 

We need to both initialize private attributes and call the user-defined model\_post\_init method.

currency: str

Three-letter ISO 4217 currency code, or 'XTR' for payments in Telegram Stars

total\_amount: int

Total price in the *smallest units* of the currency (integer, **not** float/double). For example, for a price of US\$ 1.45 pass amount = 145. See the *exp* parameter in currencies.json, it shows the number of digits past the decimal point for each currency (2 for the majority of currencies).

#### **LabeledPrice**

class aiogram.types.labeled\_price.LabeledPrice(\*, label: str, amount: int, \*\*extra\_data: Any)

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

Source: https://core.telegram.org/bots/api#labeledprice

label: str

Portion label

model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

 $model_post_init(context: Any, /) \rightarrow None$ 

We need to both initialize private attributes and call the user-defined model\_post\_init method.

amount: int

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

### OrderInfo

```
class aiogram.types.order_info.0rderInfo(*, name: str | None = None, phone_number: str | None = None,
                                                email: str | None = None, shipping_address: ShippingAddress
                                                | None = None, **extra_data: Any)
     This object represents information about an order.
     Source: https://core.telegram.org/bots/api#orderinfo
     name: str | None
          Optional. User name
     phone_number: str | None
          Optional. User's phone number
     model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
          A dictionary of computed field names and their corresponding ComputedFieldInfo objects.
     model_post_init(context: Any, /) \rightarrow None
          We need to both initialize private attributes and call the user-defined model_post_init method.
     email: str | None
          Optional. User email
     shipping_address: ShippingAddress | None
          Optional. User shipping address
```

### **PaidMediaPurchased**

This object contains information about a paid media purchase.

Source: https://core.telegram.org/bots/api#paidmediapurchased

```
from_user: User
```

User who purchased the media

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
paid_media_payload: str
```

Bot-specified paid media payload

## **PreCheckoutQuery**

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

Source: https://core.telegram.org/bots/api#precheckoutquery

id: str

Unique query identifier

from\_user: User

User who sent the query

currency: str

Three-letter ISO 4217 currency code, or 'XTR' for payments in Telegram Stars

total amount: int

Total price in the *smallest units* of the currency (integer, **not** float/double). For example, for a price of US\$ 1.45 pass amount = 145. See the *exp* parameter in currencies.json, it shows the number of digits past the decimal point for each currency (2 for the majority of currencies).

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

invoice\_payload: str

Bot-specified invoice payload

```
shipping_option_id: str | None
```

Optional. Identifier of the shipping option chosen by the user

```
order_info: OrderInfo | None
```

Optional. Order information provided by the user

**answer**(ok: bool,  $error\_message$ :  $str \mid None = None$ , \*\*kwargs: Any)  $\rightarrow AnswerPreCheckoutQuery$ 

Shortcut for method aiogram.methods.answer\_pre\_checkout\_query.AnswerPreCheckoutQuery will automatically fill method attributes:

• pre\_checkout\_query\_id

Once the user has confirmed their payment and shipping details, the Bot API sends the final confirmation in the form of an aiogram.types.update.Update with the field pre\_checkout\_query. Use this method to respond to such pre-checkout queries. On success, True is returned. Note: The Bot API must receive an answer within 10 seconds after the pre-checkout query was sent.

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

### **Parameters**

• **ok** – 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** – 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**

```
instance of method aiogram.methods.answer_pre_checkout_query.
AnswerPreCheckoutQuery
```

# RefundedPayment

This object contains basic information about a refunded payment.

Source: https://core.telegram.org/bots/api#refundedpayment

```
currency: Literal['XTR']
```

Three-letter ISO 4217 currency code, or 'XTR' for payments in Telegram Stars. Currently, always 'XTR'

```
total_amount: int
```

Total refunded price in the *smallest units* of the currency (integer, **not** float/double). For example, for a price of US\$ 1.45, total\_amount = 145. See the *exp* parameter in currencies.json, it shows the number of digits past the decimal point for each currency (2 for the majority of currencies).

```
invoice_payload: str
```

Bot-specified invoice payload

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
telegram_payment_charge_id: str
```

Telegram payment identifier

```
provider_payment_charge_id: str | None
```

Optional. Provider payment identifier

# RevenueWithdrawalState

class aiogram.types.revenue\_withdrawal\_state.RevenueWithdrawalState(\*\*extra\_data: Any)

This object describes the state of a revenue withdrawal operation. Currently, it can be one of

- aiogram.types.revenue\_withdrawal\_state\_pending.RevenueWithdrawalStatePending
- aiogram.types.revenue\_withdrawal\_state\_succeeded.RevenueWithdrawalStateSucceeded
- aiogram.types.revenue\_withdrawal\_state\_failed.RevenueWithdrawalStateFailed

Source: https://core.telegram.org/bots/api#revenuewithdrawalstate

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### RevenueWithdrawalStateFailed

```
\textbf{class} \texttt{ aiogram.types.revenue\_withdrawal\_state\_failed.} \textbf{RevenueWithdrawalStateFailed(*}, \textit{type:}
```

eral[RevenueWithdrawalState = RevenueWithdrawal-State-Type.FAILED, \*\*extra data:

Any)

The withdrawal failed and the transaction was refunded.

Source: https://core.telegram.org/bots/api#revenuewithdrawalstatefailed

type: Literal[RevenueWithdrawalStateType.FAILED]

Type of the state, always 'failed'

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

### RevenueWithdrawalStatePending

```
class aiogram.types.revenue_withdrawal_state_pending.RevenueWithdrawalStatePending(*, type:
```

```
Lit-
eral[RevenueWithdrawalSt
= Rev-
enue-
With-
drawal-
```

State-Type.PENDING,

\*\*extra data:

Any)

The withdrawal is in progress.

Source: https://core.telegram.org/bots/api#revenuewithdrawalstatepending

type: Literal[RevenueWithdrawalStateType.PENDING]

Type of the state, always 'pending'

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### RevenueWithdrawalStateSucceeded

class aiogram.types.revenue\_withdrawal\_state\_succeeded.RevenueWithdrawalStateSucceeded(\*,

```
type:
Lit-
eral[RevenueWithdra
Rev-
enue-
With-
drawal-
State-
Type.SUCCEEDED,
date:
date-
time serializer,
re-
turn_type=int,
when_used=unless
none)],
url:
str,
**ex-
tra_data:
```

Any)

The withdrawal succeeded.

Source: https://core.telegram.org/bots/api#revenuewithdrawalstatesucceeded

# type: Literal[RevenueWithdrawalStateType.SUCCEEDED]

Type of the state, always 'succeeded'

date: DateTime

Date the withdrawal was completed in Unix time

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### url: str

An HTTPS URL that can be used to see transaction details

# **ShippingAddress**

```
class aiogram.types.shipping_address.ShippingAddress(*, country_code: str, state: str, city: str,
                                                                street_line1: str, street_line2: str, post_code:
                                                                str, **extra_data: Any)
     This object represents a shipping address.
     Source: https://core.telegram.org/bots/api#shippingaddress
     country_code: str
          Two-letter ISO 3166-1 alpha-2 country code
     state: str
          State, if applicable
     city: str
           City
     model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
           A dictionary of computed field names and their corresponding ComputedFieldInfo objects.
     model_post_init(context: Any, /) \rightarrow None
           We need to both initialize private attributes and call the user-defined model_post_init method.
     street_line1: str
           First line for the address
     street_line2: str
           Second line for the address
     post_code: str
           Address post code
ShippingOption
class aiogram.types.shipping_option.ShippingOption(*, id: str, title: str, prices: List[LabeledPrice],
                                                              **extra data: Any)
     This object represents one shipping option.
     Source: https://core.telegram.org/bots/api#shippingoption
     id: str
           Shipping option identifier
     title: str
           Option title
     model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
           A dictionary of computed field names and their corresponding ComputedFieldInfo objects.
     model_post_init(context: Any, /) \rightarrow None
           We need to both initialize private attributes and call the user-defined model_post_init method.
     prices: List[LabeledPrice]
           List of price portions
```

# **ShippingQuery**

This object contains information about an incoming shipping query.

Source: https://core.telegram.org/bots/api#shippingquery

#### id: str

Unique query identifier

from\_user: User

User who sent the query

invoice\_payload: str

Bot-specified invoice payload

model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
shipping_address: ShippingAddress
```

User specified shipping address

**answer**(ok: bool, shipping\_options: List[ShippingOption] | None = None, error\_message: str | None = None, \*\*kwargs: Any) → AnswerShippingQuery

 $Shortcut\ for\ method\ aiogram.methods.answer\_shipping\_query. AnswerShippingQuery\ will\ automatically\ fill\ method\ attributes:$ 

• shipping\_query\_id

If you sent an invoice requesting a shipping address and the parameter *is\_flexible* was specified, the Bot API will send an *aiogram.types.update.Update* with a *shipping\_query* field to the bot. Use this method to reply to shipping queries. On success, True is returned.

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

#### **Parameters**

- **ok** Pass 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** Required if *ok* is True. A JSON-serialized array of available shipping options.
- **error\_message** 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

 $\begin{array}{ll} \text{instance} & \text{of} & \text{method} & \textit{aiogram.methods.answer\_shipping\_query}. \\ & \textit{AnswerShippingQuery} \end{array}$ 

### **StarTransaction**

Describes a Telegram Star transaction.

Source: https://core.telegram.org/bots/api#startransaction

#### id: str

Unique identifier of the transaction. Coincides with the identifier of the original transaction for refund transactions. Coincides with *SuccessfulPayment.telegram\_payment\_charge\_id* for successful incoming payments from users.

#### amount: int

Number of Telegram Stars transferred by the transaction

#### date: DateTime

Date the transaction was created in Unix time

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
source: TransactionPartnerUser | TransactionPartnerFragment |
TransactionPartnerTelegramAds | TransactionPartnerOther | None
```

*Optional*. Source of an incoming transaction (e.g., a user purchasing goods or services, Fragment refunding a failed withdrawal). Only for incoming transactions

```
receiver: TransactionPartnerUser | TransactionPartnerFragment |
TransactionPartnerTelegramAds | TransactionPartnerOther | None
```

Optional. Receiver of an outgoing transaction (e.g., a user for a purchase refund, Fragment for a withdrawal). Only for outgoing transactions

#### **StarTransactions**

Contains a list of Telegram Star transactions.

Source: https://core.telegram.org/bots/api#startransactions

transactions: List[StarTransaction]

The list of transactions

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

## SuccessfulPayment

This object contains basic information about a successful payment.

Source: https://core.telegram.org/bots/api#successfulpayment

```
currency: str
```

Three-letter ISO 4217 currency code, or 'XTR' for payments in Telegram Stars

```
total_amount: int
```

Total price in the *smallest units* of the currency (integer, **not** float/double). For example, for a price of US\$ 1.45 pass amount = 145. See the *exp* parameter in currencies.json, it shows the number of digits past the decimal point for each currency (2 for the majority of currencies).

```
invoice_payload: str
```

Bot-specified invoice payload

### telegram\_payment\_charge\_id: str

Telegram payment identifier

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
provider_payment_charge_id: str
```

Provider payment identifier

```
shipping_option_id: str | None
```

Optional. Identifier of the shipping option chosen by the user

```
order_info: OrderInfo | None
```

Optional. Order information provided by the user

### **TransactionPartner**

```
class aiogram.types.transaction_partner.TransactionPartner(**extra_data: Any)
```

This object describes the source of a transaction, or its recipient for outgoing transactions. Currently, it can be one of

- aiogram.types.transaction\_partner\_user.TransactionPartnerUser
- aiogram.types.transaction\_partner\_fragment.TransactionPartnerFragment
- aiogram.types.transaction\_partner\_telegram\_ads.TransactionPartnerTelegramAds
- aiogram.types.transaction\_partner\_other.TransactionPartnerOther

Source: https://core.telegram.org/bots/api#transactionpartner

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

# **TransactionPartnerFragment**

```
class aiogram.types.transaction_partner_fragment.TransactionPartnerFragment(*, type: Lit-
```

eral[TransactionPartnerType.FRAG

= Transaction-

Partner-

Type.FRAGMENT,

with-

drawal\_state:

RevenueWith-

drawalStatePend-

ing

RevenueWith-

drawalStateSuc-

ceeded

RevenueWith-

drawalState-

Failed | None =

None,

\*\*extra\_data:

Any)

Describes a withdrawal transaction with Fragment.

Source: https://core.telegram.org/bots/api#transactionpartnerfragment

## type: Literal[TransactionPartnerType.FRAGMENT]

Type of the transaction partner, always 'fragment'

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model post init method.

```
withdrawal_state: RevenueWithdrawalStatePending | RevenueWithdrawalStateSucceeded |
RevenueWithdrawalStateFailed | None
```

Optional. State of the transaction if the transaction is outgoing

#### **TransactionPartnerOther**

Describes a transaction with an unknown source or recipient.

Source: https://core.telegram.org/bots/api#transactionpartnerother

type: Literal[TransactionPartnerType.OTHER]

Type of the transaction partner, always 'other'

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

# **TransactionPartnerTelegramAds**

Describes a withdrawal transaction to the Telegram Ads platform.

Source: https://core.telegram.org/bots/api#transactionpartnertelegramads

```
type: Literal[TransactionPartnerType.TELEGRAM_ADS]
```

Type of the transaction partner, always 'telegram\_ads'

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

### **TransactionPartnerUser**

```
class aiogram.types.transaction_partner_user.TransactionPartnerUser(*, type: Lit-
                                                                                eral[TransactionPartnerType.USER]
                                                                                 = TransactionPartner-
                                                                                 Type. USER, user: User,
                                                                                invoice_payload: str | None
                                                                                = None, paid media:
                                                                                List/PaidMediaPreview |
                                                                                PaidMediaPhoto I
                                                                                PaidMediaVideo | | None =
                                                                                None.
                                                                                paid_media_payload: str |
                                                                                None = None,
                                                                                 **extra data: Any)
     Describes a transaction with a user.
     Source: https://core.telegram.org/bots/api#transactionpartneruser
     type: Literal[TransactionPartnerType.USER]
          Type of the transaction partner, always 'user'
     user: User
          Information about the user
     invoice_payload: str | None
          Optional. Bot-specified invoice payload
     model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
          A dictionary of computed field names and their corresponding ComputedFieldInfo objects.
     model_post_init(context: Any, /) \rightarrow None
          We need to both initialize private attributes and call the user-defined model_post_init method.
     paid_media: List[PaidMediaPreview | PaidMediaPhoto | PaidMediaVideo] | None
          Optional. Information about the paid media bought by the user
     paid_media_payload: str | None
          Optional. Bot-specified paid media payload
```

### **Telegram Passport**

#### **EncryptedCredentials**

Describes data required for decrypting and authenticating <code>aiogram.types.encrypted\_passport\_element.</code> EncryptedPassportElement. See the Telegram Passport Documentation for a complete description of the data decryption and authentication processes.

Source: https://core.telegram.org/bots/api#encryptedcredentials

#### data: str

Base64-encoded encrypted JSON-serialized data with unique user's payload, data hashes and secrets required for aiogram.types.encrypted\_passport\_element.EncryptedPassportElement decryption and authentication

#### hash: str

Base64-encoded data hash for data authentication

# model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### secret: str

Base64-encoded secret, encrypted with the bot's public RSA key, required for data decryption

# **EncryptedPassportElement**

class aiogram.types.encrypted\_passport\_element.EncryptedPassportElement(\*, type: str, hash: str,

data: str | None = *None*, *phone\_number*:  $str \mid None = None$ . email: str | None = None, files: List[PassportFile] | None = None,front side: PassportFile | *None* = None, reverse side: PassportFile | *None* = None, selfie: PassportFile | None = None, translation: *List*/PassportFile/ None = None, \*\*extra\_data: Any)

Describes documents or other Telegram Passport elements shared with the bot by the user.

Source: https://core.telegram.org/bots/api#encryptedpassportelement

# type: str

Element type. One of 'personal\_details', 'passport', 'driver\_license', 'identity\_card', 'internal\_passport', 'address', 'utility\_bill', 'bank\_statement', 'rental\_agreement', 'passport\_registration', 'temporary\_registration', 'phone\_number', 'email'.

#### hash: str

Base64-encoded element hash for using in aiogram.types.passport\_element\_error\_unspecified. PassportElementErrorUnspecified

### data: str | None

Optional. Base64-encoded encrypted Telegram Passport element data provided by the user; available only for 'personal\_details', 'passport', 'driver\_license', 'identity\_card', 'internal\_passport' and 'address' types. Can be decrypted and verified using the accompanying aiogram.types.encrypted\_credentials. EncryptedCredentials.

### phone\_number: str | None

Optional. User's verified phone number; available only for 'phone\_number' type

### email: str | None

Optional. User's verified email address; available only for 'email' type

# model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### files: List[PassportFile] | None

Optional. Array of encrypted files with documents provided by the user; available only for 'utility\_bill', 'bank\_statement', 'rental\_agreement', 'passport\_registration' and 'temporary\_registration' types. Files can be decrypted and verified using the accompanying aiogram.types.encrypted\_credentials. EncryptedCredentials.

### front\_side: PassportFile | None

*Optional*. Encrypted file with the front side of the document, provided by the user; available only for 'passport', 'driver\_license', 'identity\_card' and 'internal\_passport'. The file can be decrypted and verified using the accompanying aiogram.types.encrypted\_credentials.EncryptedCredentials.

## reverse\_side: PassportFile | None

*Optional*. Encrypted file with the reverse side of the document, provided by the user; available only for 'driver\_license' and 'identity\_card'. The file can be decrypted and verified using the accompanying <code>aiogram.types.encrypted\_credentials.EncryptedCredentials</code>.

### selfie: PassportFile | None

*Optional.* Encrypted file with the selfie of the user holding a document, provided by the user; available if requested for 'passport', 'driver\_license', 'identity\_card' and 'internal\_passport'. The file can be decrypted and verified using the accompanying aiogram.types.encrypted\_credentials. EncryptedCredentials.

## translation: List[PassportFile] | None

Optional. Array of encrypted files with translated versions of documents provided by the user; available if requested for 'passport', 'driver\_license', 'identity\_card', 'internal\_passport', 'utility\_bill', 'bank\_statement', 'rental\_agreement', 'passport\_registration' and 'temporary\_registration' types. Files can be decrypted and verified using the accompanying <code>aiogram.types.encrypted\_credentials.EncryptedCredentials.</code>

### **PassportData**

Describes Telegram Passport data shared with the bot by the user.

Source: https://core.telegram.org/bots/api#passportdata

## data: List[EncryptedPassportElement]

Array with information about documents and other Telegram Passport elements that was shared with the

### model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

### credentials: EncryptedCredentials

Encrypted credentials required to decrypt the data

# **PassportElementError**

```
class aiogram.types.passport_element_error.PassportElementError(**extra_data: Any)
```

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

- aiogram.types.passport\_element\_error\_data\_field.PassportElementErrorDataField
- aiogram.types.passport\_element\_error\_front\_side.PassportElementErrorFrontSide
- aiogram.types.passport\_element\_error\_reverse\_side.PassportElementErrorReverseSide
- aiogram.types.passport\_element\_error\_selfie.PassportElementErrorSelfie
- aiogram.types.passport\_element\_error\_file.PassportElementErrorFile
- aiogram.types.passport\_element\_error\_files.PassportElementErrorFiles
- aiogram.types.passport\_element\_error\_translation\_file.PassportElementErrorTranslationFile
- aiogram.types.passport\_element\_error\_translation\_files.PassportElementErrorTranslationFiles
- aiogram.types.passport\_element\_error\_unspecified.PassportElementErrorUnspecified

Source: https://core.telegram.org/bots/api#passportelementerror

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

# **PassportElementErrorDataField**

class aiogram.types.passport\_element\_error\_data\_field.PassportElementErrorDataField(\*,

```
source:
Lit-
eral[PassportElementErr
Pass-
portEle-
mentEr-
rorType.DATA,
type:
str,
field_name:
str.
data_hash:
str.
mes-
sage:
str.
**ex-
tra data:
Any)
```

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.

Source: https://core.telegram.org/bots/api#passportelementerrordatafield

# source: Literal[PassportElementErrorType.DATA]

Error source, must be data

#### type: str

The section of the user's Telegram Passport which has the error, one of 'personal\_details', 'passport', 'driver\_license', 'identity\_card', 'internal\_passport', 'address'

#### field\_name: str

Name of the data field which has the error

# model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

#### $model_post_init(context: Any, /) \rightarrow None$

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### data\_hash: str

Base64-encoded data hash

# message: str

Error message

# **PassportElementErrorFile**

class aiogram.types.passport\_element\_error\_file.PassportElementErrorFile(\*, source: Lit-

eral[PassportElementErrorType.FILE]
=
PassportElementEr-

rorType.FILE, type: str, file\_hash: str, message: str, \*\*extra\_data: Any)

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

Source: https://core.telegram.org/bots/api#passportelementerrorfile

#### source: Literal[PassportElementErrorType.FILE]

Error source, must be file

### type: str

The section of the user's Telegram Passport which has the issue, one of 'utility\_bill', 'bank\_statement', 'rental\_agreement', 'passport\_registration', 'temporary\_registration'

## model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

### $model_post_init(context: Any, /) \rightarrow None$

We need to both initialize private attributes and call the user-defined model\_post\_init method.

file\_hash: str

Base64-encoded file hash

message: str Error message

## **PassportElementErrorFiles**

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

Source: https://core.telegram.org/bots/api#passportelementerrorfiles

# source: Literal[PassportElementErrorType.FILES]

Error source, must be files

type: str

The section of the user's Telegram Passport which has the issue, one of 'utility\_bill', 'bank\_statement', 'rental\_agreement', 'passport\_registration', 'temporary\_registration'

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

file\_hashes: List[str]

List of base64-encoded file hashes

message: str Error message

# **PassportElementErrorFrontSide**

Any)

class aiogram.types.passport\_element\_error\_front\_side.PassportElementErrorFrontSide(\*,

```
source:
Lit-
eral[PassportElementErr
Pass-
portEle-
mentEr-
rorType.FRONT_SIDE,
type:
str,
file_hash:
str,
mes-
sage:
str,
**ex-
tra data:
Any)
```

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.

Source: https://core.telegram.org/bots/api#passportelementerrorfrontside

## source: Literal[PassportElementErrorType.FRONT\_SIDE]

Error source, must be front\_side

#### type: str

The section of the user's Telegram Passport which has the issue, one of 'passport', 'driver\_license', 'identity\_card', 'internal\_passport'

## model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

## $model_post_init(context: Any, /) \rightarrow None$

We need to both initialize private attributes and call the user-defined model\_post\_init method.

# file\_hash: str

Base64-encoded hash of the file with the front side of the document

# message: str Error message

## **PassportElementErrorReverseSide**

class aiogram.types.passport\_element\_error\_reverse\_side.PassportElementErrorReverseSide(\*,

```
Lit-
eral[PassportEleme
Pass-
portEle-
mentEr-
rorType.REVERSE_
type:
str,
file_hash:
str,
mes-
sage:
str,
**ex-
tra data:
Any)
```

source:

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.

Source: https://core.telegram.org/bots/api#passportelementerrorreverseside

## source: Literal[PassportElementErrorType.REVERSE\_SIDE]

Error source, must be reverse\_side

type: str

The section of the user's Telegram Passport which has the issue, one of 'driver\_license', 'identity\_card'

## model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

## file\_hash: str

Base64-encoded hash of the file with the reverse side of the document

## message: str

Error message

## **PassportElementErrorSelfie**

Any)

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

Source: https://core.telegram.org/bots/api#passportelementerrorselfie

## source: Literal[PassportElementErrorType.SELFIE]

Error source, must be selfie

#### type: str

The section of the user's Telegram Passport which has the issue, one of 'passport', 'driver\_license', 'identity\_card', 'internal\_passport'

## model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### file\_hash: str

Base64-encoded hash of the file with the selfie

# message: str Error message

## **PassportElementErrorTranslationFile**

class aiogram.types.passport\_element\_error\_translation\_file.PassportElementErrorTranslationFile(\*,

```
Pass-
portEle-
mentEr-
rorType.T
type:
str,
file_hash.
str,
mes-
sage:
```

str,
\*\*extra\_data:
Any)

source: Literal[Pass

Represents an issue with one of the files that constitute the translation of a document. The error is considered resolved when the file changes.

Source: https://core.telegram.org/bots/api#passportelementerrortranslationfile

## source: Literal[PassportElementErrorType.TRANSLATION\_FILE]

Error source, must be translation\_file

#### type: str

Type of element of the user's Telegram Passport which has the issue, one of 'passport', 'driver\_license', 'identity\_card', 'internal\_passport', 'utility\_bill', 'bank\_statement', 'rental\_agreement', 'passport\_registration', 'temporary\_registration'

## model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

#### $model_post_init(context: Any, /) \rightarrow None$

We need to both initialize private attributes and call the user-defined model\_post\_init method.

## file\_hash: str

Base64-encoded file hash

message: str
Error message

## **PassportElementErrorTranslationFiles**

class aiogram.types.passport\_element\_error\_translation\_files.PassportElementErrorTranslationFiles(\*,

=
PassportEld
mentEld
rorTyp
type:
str,
file\_had
List[stt]

sage: str, \*\*extra\_da Any)

source Literal[Pa

Represents an issue with the translated version of a document. The error is considered resolved when a file with the document translation change.

**Source:** https://core.telegram.org/bots/api#passportelementerrortranslationfiles

## source: Literal[PassportElementErrorType.TRANSLATION\_FILES]

Error source, must be translation\_files

## type: str

Type of element of the user's Telegram Passport which has the issue, one of 'passport', 'driver\_license', 'identity\_card', 'internal\_passport', 'utility\_bill', 'bank\_statement', 'rental\_agreement', 'passport\_registration', 'temporary\_registration'

## model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

## file\_hashes: List[str]

List of base64-encoded file hashes

message: str Error message

## **PassportElementErrorUnspecified**

```
class aiogram.types.passport_element_error_unspecified.PassportElementErrorUnspecified(*,
```

```
source:
Lit-
eral[PassportElemen
Pass-
portEle-
mentEr-
rorType.UNSPECIFI
type:
str,
el-
ment hash:
str,
mes-
sage:
str,
**ex-
tra_data:
Any)
```

Represents an issue in an unspecified place. The error is considered resolved when new data is added.

Source: https://core.telegram.org/bots/api#passportelementerrorunspecified

## source: Literal[PassportElementErrorType.UNSPECIFIED]

Error source, must be unspecified

# type: str

Type of element of the user's Telegram Passport which has the issue

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### element hash: str

Base64-encoded element hash

# message: str Error message

## **PassportFile**

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.

Source: https://core.telegram.org/bots/api#passportfile

#### file\_id: str

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

# file\_unique\_id: str

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

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

**file\_size:** int
File size in bytes

file\_date: DateTime

Unix time when the file was uploaded

## **Getting updates**

## **Update**

```
class aiogram.types.update(*, update_id: int, message: Message | None = None, edited_message:
                                        Message | None = None, channel post: Message | None = None,
                                        edited_channel_post: Message | None = None, business_connection:
                                        BusinessConnection | None = None, business message: Message | None
                                        = None, edited_business_message: Message | None = None,
                                        deleted business messages: BusinessMessagesDeleted | None = None,
                                        message_reaction: MessageReactionUpdated | None = None,
                                        message reaction count: MessageReactionCountUpdated | None =
                                        None, inline_query: InlineQuery | None = None, chosen_inline_result:
                                        ChosenInlineResult | None = None, callback_query: CallbackQuery |
                                        None = None, shipping_query: ShippingQuery | None = None,
                                        pre_checkout_query: PreCheckoutQuery | None = None,
                                        purchased_paid_media: PaidMediaPurchased | None = None, poll: Poll
                                        | None = None, poll\_answer: PollAnswer | None = None,
                                        my_chat_member: ChatMemberUpdated | None = None, chat_member:
                                        ChatMemberUpdated | None = None, chat_join_request:
                                        ChatJoinRequest | None = None, chat_boost: ChatBoostUpdated | None
                                        = None, removed_chat_boost: ChatBoostRemoved | None = None,
                                        **extra data: Any)
```

This object represents an incoming update.

At most **one** of the optional parameters can be present in any given update.

Source: https://core.telegram.org/bots/api#update

## update\_id: int

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

#### message: Message | None

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

```
edited_message: Message | None
```

*Optional*. New version of a message that is known to the bot and was edited. This update may at times be triggered by changes to message fields that are either unavailable or not actively used by your bot.

```
channel_post: Message | None
```

Optional. New incoming channel post of any kind - text, photo, sticker, etc.

```
edited_channel_post: Message | None
```

*Optional*. New version of a channel post that is known to the bot and was edited. This update may at times be triggered by changes to message fields that are either unavailable or not actively used by your bot.

```
business_connection: BusinessConnection | None
```

Optional. The bot was connected to or disconnected from a business account, or a user edited an existing connection with the bot

```
business_message: Message | None
```

Optional. New message from a connected business account

```
edited_business_message: Message | None
```

Optional. New version of a message from a connected business account

## deleted\_business\_messages: BusinessMessagesDeleted | None

Optional. Messages were deleted from a connected business account

```
message_reaction: MessageReactionUpdated | None
```

*Optional*. A reaction to a message was changed by a user. The bot must be an administrator in the chat and must explicitly specify "message\_reaction" in the list of *allowed\_updates* to receive these updates. The update isn't received for reactions set by bots.

```
message_reaction_count: MessageReactionCountUpdated | None
```

*Optional*. Reactions to a message with anonymous reactions were changed. The bot must be an administrator in the chat and must explicitly specify "message\_reaction\_count" in the list of *allowed\_updates* to receive these updates. The updates are grouped and can be sent with delay up to a few minutes.

```
inline_query: InlineQuery | None
```

Optional. New incoming inline query

```
chosen_inline_result: ChosenInlineResult | None
```

*Optional*. The result of an inline query that was chosen by a user and sent to their chat partner. Please see our documentation on the feedback collecting for details on how to enable these updates for your bot.

```
callback_query: CallbackQuery | None
```

Optional. New incoming callback query

### model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

 $model_post_init(context: Any, /) \rightarrow None$ 

We need to both initialize private attributes and call the user-defined model\_post\_init method.

shipping\_query: ShippingQuery | None

Optional. New incoming shipping query. Only for invoices with flexible price

pre\_checkout\_query: PreCheckoutQuery | None

Optional. New incoming pre-checkout query. Contains full information about checkout

purchased\_paid\_media: PaidMediaPurchased | None

Optional. A user purchased paid media with a non-empty payload sent by the bot in a non-channel chat

poll: Poll | None

*Optional*. New poll state. Bots receive only updates about manually stopped polls and polls, which are sent by the bot

poll\_answer: PollAnswer | None

*Optional*. A user changed their answer in a non-anonymous poll. Bots receive new votes only in polls that were sent by the bot itself.

my\_chat\_member: ChatMemberUpdated | None

*Optional*. The bot's chat member status was updated in a chat. For private chats, this update is received only when the bot is blocked or unblocked by the user.

chat\_member: ChatMemberUpdated | None

*Optional*. A chat member's status was updated in a chat. The bot must be an administrator in the chat and must explicitly specify "chat\_member" in the list of *allowed\_updates* to receive these updates.

chat\_join\_request: ChatJoinRequest | None

*Optional*. A request to join the chat has been sent. The bot must have the *can\_invite\_users* administrator right in the chat to receive these updates.

chat\_boost: ChatBoostUpdated | None

*Optional*. A chat boost was added or changed. The bot must be an administrator in the chat to receive these updates.

removed\_chat\_boost: ChatBoostRemoved | None

*Optional*. A boost was removed from a chat. The bot must be an administrator in the chat to receive these updates.

property event\_type: str

Detect update type If update type is unknown, raise UpdateTypeLookupError

Returns

property event: TelegramObject

exception aiogram.types.update.UpdateTypeLookupError

Update does not contain any known event type.

## WebhookInfo

class aiogram.types.webhook\_info.WebhookInfo(\*, url: str, has\_custom\_certificate: bool,

pending\_update\_count: int, ip\_address: str | None = None, last\_error\_date: \_datetime\_serializer, return\_type=int, when\_used=unless - none)] | None = None, last\_error\_message: str | None = None, last\_synchronization\_error\_date: \_datetime\_serializer, return\_type=int, when\_used=unless - none)] | None = None, max\_connections: int | None = None, allowed\_updates: List[str] | None = None, \*\*extra\_data: Any)

Describes the current status of a webhook.

Source: https://core.telegram.org/bots/api#webhookinfo

url: str

Webhook URL, may be empty if webhook is not set up

has\_custom\_certificate: bool

True, if a custom certificate was provided for webhook certificate checks

pending\_update\_count: int

Number of updates awaiting delivery

ip\_address: str | None

Optional. Currently used webhook IP address

last\_error\_date: DateTime | None

Optional. Unix time for the most recent error that happened when trying to deliver an update via webhook

model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

 $model_post_init(context: Any, /) \rightarrow None$ 

We need to both initialize private attributes and call the user-defined model\_post\_init method.

last\_error\_message: str | None

*Optional*. Error message in human-readable format for the most recent error that happened when trying to deliver an update via webhook

last\_synchronization\_error\_date: DateTime | None

*Optional*. Unix time of the most recent error that happened when trying to synchronize available updates with Telegram datacenters

max\_connections: int | None

*Optional*. The maximum allowed number of simultaneous HTTPS connections to the webhook for update delivery

allowed\_updates: List[str] | None

Optional. A list of update types the bot is subscribed to. Defaults to all update types except chat\_member

#### **Games**

#### **CallbackGame**

```
class aiogram.types.callback_game.CallbackGame(**extra_data: Any)
```

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

Source: https://core.telegram.org/bots/api#callbackgame

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### Game

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

Source: https://core.telegram.org/bots/api#game

title: str

Title of the game

description: str

Description of the game

photo: List[PhotoSize]

Photo that will be displayed in the game message in chats.

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
text: str | None
```

Optional. Brief description of the game or high scores included in the game message. Can be automatically edited to include current high scores for the game when the bot calls <code>aiogram.methods.set\_game\_score.SetGameScore</code>, or manually edited using <code>aiogram.methods.edit\_message\_text.EditMessageText</code>. 0-4096 characters.

```
text_entities: List[MessageEntity] | None
```

Optional. Special entities that appear in text, such as usernames, URLs, bot commands, etc.

```
animation: Animation | None
```

Optional. Animation that will be displayed in the game message in chats. Upload via BotFather

## **GameHighScore**

#### 2.3.4 Methods

Score

Here is list of all available API methods:

#### **Stickers**

#### addStickerToSet

```
Returns: bool
```

Use this method to add a new sticker to a set created by the bot. Emoji sticker sets can have up to 200 stickers. Other sticker sets can have up to 120 stickers. Returns True on success.

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

```
user_id: int

User identifier of sticker set owner

name: str

Sticker set name
```

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### sticker: InputSticker

A JSON-serialized object with information about the added sticker. If exactly the same sticker had already been added to the set, then the set isn't changed.

## **Usage**

## As bot method

```
result: bool = await bot.add_sticker_to_set(...)
```

## Method as object

Imports:

- from aiogram.methods.add\_sticker\_to\_set import AddStickerToSet
- alias: from aiogram.methods import AddStickerToSet

## With specific bot

```
result: bool = await bot(AddStickerToSet(...))
```

# As reply into Webhook in handler

```
return AddStickerToSet(...)
```

#### createNewStickerSet

Returns: bool

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. Returns True on success.

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

## user\_id: int

User identifier of created sticker set owner

#### name: str

Short name of sticker set, to be used in t.me/addstickers/ URLs (e.g., *animals*). Can contain only English letters, digits and underscores. Must begin with a letter, can't contain consecutive underscores and must end in "\_by\_<bot\_username>". <bot\_username> is case insensitive. 1-64 characters.

#### title: str

Sticker set title, 1-64 characters

#### stickers: List[InputSticker]

A JSON-serialized list of 1-50 initial stickers to be added to the sticker set

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
sticker_type: str | None
```

Type of stickers in the set, pass 'regular', 'mask', or 'custom\_emoji'. By default, a regular sticker set is created.

```
needs_repainting: bool | None
```

Pass True if stickers in the sticker set must be repainted to the color of text when used in messages, the accent color if used as emoji status, white on chat photos, or another appropriate color based on context; for custom emoji sticker sets only

```
sticker_format: str | None
```

Format of stickers in the set, must be one of 'static', 'animated', 'video'

Deprecated since version API:7.2: https://core.telegram.org/bots/api-changelog#march-31-2024

#### **Usage**

#### As bot method

```
result: bool = await bot.create_new_sticker_set(...)
```

## Method as object

## Imports:

- $\bullet \ \, from \ \, aiogram.methods.create\_new\_sticker\_set \ \, import \ \, CreateNewStickerSet$
- $\bullet$  alias: from alogram.methods import CreateNewStickerSet

```
result: bool = await bot(CreateNewStickerSet(...))
```

# As reply into Webhook in handler

```
return CreateNewStickerSet(...)
```

#### deleteStickerFromSet

Returns: bool

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

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

sticker: str

File identifier of the sticker

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

# **Usage**

## As bot method

```
result: bool = await bot.delete_sticker_from_set(...)
```

# Method as object

Imports:

- from aiogram.methods.delete\_sticker\_from\_set import DeleteStickerFromSet
- alias: from aiogram.methods import DeleteStickerFromSet

```
result: bool = await bot(DeleteStickerFromSet(...))
```

# As reply into Webhook in handler

```
return DeleteStickerFromSet(...)
```

## As shortcut from received object

• aiogram.types.sticker.Sticker.delete\_from\_set()

#### deleteStickerSet

Returns: bool

class aiogram.methods.delete\_sticker\_set.DeleteStickerSet(\*, name: str, \*\*extra\_data: Any)

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

Source: https://core.telegram.org/bots/api#deletestickerset

name: str

Sticker set name

model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

 $model_post_init(context: Any, /) \rightarrow None$ 

We need to both initialize private attributes and call the user-defined model\_post\_init method.

## **Usage**

#### As bot method

```
result: bool = await bot.delete_sticker_set(...)
```

## Method as object

#### Imports:

- from aiogram.methods.delete\_sticker\_set import DeleteStickerSet
- alias: from aiogram.methods import DeleteStickerSet

```
result: bool = await bot(DeleteStickerSet(...))
```

# As reply into Webhook in handler

```
return DeleteStickerSet(...)
```

## getCustomEmojiStickers

```
Returns: List[Sticker]
```

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

Source: https://core.telegram.org/bots/api#getcustomemojistickers

```
custom_emoji_ids: List[str]
```

A JSON-serialized list of custom emoji identifiers. At most 200 custom emoji identifiers can be specified.

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

## **Usage**

# As bot method

```
result: List[Sticker] = await bot.get_custom_emoji_stickers(...)
```

# Method as object

# Imports:

- $\bullet \ from \ aiogram.methods.get\_custom\_emoji\_stickers \ import \ GetCustomEmojiStickers$
- alias: from aiogram.methods import GetCustomEmojiStickers

```
result: List[Sticker] = await bot(GetCustomEmojiStickers(...))
```

## getStickerSet

Returns: StickerSet

class aiogram.methods.get\_sticker\_set.GetStickerSet(\*, name: str, \*\*extra\_data: Any)

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

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

name: str

Name of the sticker set

model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

 $model_post_init(context: Any, /) \rightarrow None$ 

We need to both initialize private attributes and call the user-defined model\_post\_init method.

## **Usage**

#### As bot method

```
result: StickerSet = await bot.get_sticker_set(...)
```

## Method as object

Imports:

- from aiogram.methods.get\_sticker\_set import GetStickerSet
- alias: from aiogram.methods import GetStickerSet

## With specific bot

```
result: StickerSet = await bot(GetStickerSet(...))
```

## replaceStickerInSet

Returns: bool

Use this method to replace an existing sticker in a sticker set with a new one. The method is equivalent to calling aiogram.methods.delete\_sticker\_from\_set.DeleteStickerFromSet, then aiogram.methods.add\_sticker\_to\_set.AddStickerToSet, then aiogram.methods.set\_sticker\_position\_in\_set. SetStickerPositionInSet. Returns True on success.

Source: https://core.telegram.org/bots/api#replacestickerinset

## user\_id: int

User identifier of the sticker set owner

name: str

Sticker set name

## model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

old\_sticker: str

File identifier of the replaced sticker

sticker: InputSticker

A JSON-serialized object with information about the added sticker. If exactly the same sticker had already been added to the set, then the set remains unchanged.

## **Usage**

#### As bot method

```
result: bool = await bot.replace_sticker_in_set(...)
```

## Method as object

Imports:

- from aiogram.methods.replace\_sticker\_in\_set import ReplaceStickerInSet
- alias: from aiogram.methods import ReplaceStickerInSet

```
result: bool = await bot(ReplaceStickerInSet(...))
```

# As reply into Webhook in handler

```
return ReplaceStickerInSet(...)
```

#### sendSticker

Returns: Message

```
class aiogram.methods.send_sticker.SendSticker(*, chat_id: int | str, sticker:
```

~aiogram.types.input\_file.InputFile | str, business connection id:  $str \mid None = None$ , message\_thread\_id: int | None = None, emoji: str | *None* = *None*, *disable\_notification: bool* | *None* = None, protect\_content: bool | ~aiogram.client.default.Default | None = <Default('protect\_content')>, message\_effect\_id: str |  $None = None, reply\_parameters:$ ~aiogram.types.reply\_parameters.ReplyParameters |  $None = None, reply_markup$ : ~aiogram.types.inline\_keyboard\_markup.InlineKeyboardMarkup ~aiogram.types.reply keyboard markup.ReplyKeyboardMarkup ~aiogram.types.reply\_keyboard\_remove.ReplyKeyboardRemove | ~aiogram.types.force\_reply.ForceReply | None = None, allow sending without reply: bool | None = *None*, *reply\_to\_message\_id*: *int* | *None* = *None*, \*\*extra\_data: ~typing.Any)

Use this method to send static .WEBP, animated .TGS, or video .WEBM stickers. On success, the sent aiogram. types.message is returned.

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

## chat\_id: int | str

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

```
sticker: InputFile | str
```

Sticker to send. Pass a file\_id as String to send a file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a .WEBP sticker from the Internet, or upload a new .WEBP, .TGS, or .WEBM sticker using multipart/form-data. *More information on Sending Files* ». Video and animated stickers can't be sent via an HTTP URL.

#### business\_connection\_id: str | None

Unique identifier of the business connection on behalf of which the message will be sent

# message\_thread\_id: int | None

Unique identifier for the target message thread (topic) of the forum; for forum supergroups only

#### emoji: str | None

Emoji associated with the sticker; only for just uploaded stickers

## disable\_notification: bool | None

Sends the message silently. Users will receive a notification with no sound.

## model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
protect_content: bool | Default | None
```

Protects the contents of the sent message from forwarding and saving

```
message_effect_id: str | None
```

Unique identifier of the message effect to be added to the message; for private chats only

```
reply_parameters: ReplyParameters | None
```

Description of the message to reply to

```
reply_markup: InlineKeyboardMarkup | ReplyKeyboardMarkup | ReplyKeyboardMemove |
ForceReply | None
```

Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

```
allow_sending_without_reply: bool | None
```

Pass True if the message should be sent even if the specified replied-to message is not found

Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023

```
reply_to_message_id: int | None
```

If the message is a reply, ID of the original message

Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023

## **Usage**

## As bot method

```
result: Message = await bot.send_sticker(...)
```

# Method as object

## Imports:

- from aiogram.methods.send\_sticker import SendSticker
- alias: from aiogram.methods import SendSticker

```
result: Message = await bot(SendSticker(...))
```

# As reply into Webhook in handler

```
return SendSticker(...)
```

## As shortcut from received object

```
• aiogram.types.message.Message.answer_sticker()
```

- aiogram.types.message.Message.reply\_sticker()
- aiogram.types.chat\_join\_request.ChatJoinRequest.answer\_sticker()
- aiogram.types.chat\_join\_request.ChatJoinRequest.answer\_sticker\_pm()
- aiogram.types.chat\_member\_updated.ChatMemberUpdated.answer\_sticker()
- aiogram.types.inaccessible\_message.InaccessibleMessage.answer\_sticker()
- aiogram.types.inaccessible\_message.InaccessibleMessage.reply\_sticker()

## setCustomEmojiStickerSetThumbnail

```
Returns: bool
```

```
{\bf class} \ {\bf aiogram.methods.set\_custom\_emoji\_sticker\_set\_thumbnail.Set {\bf CustomEmojiStickerSetThumbnail} (*, note that the context of t
```

```
name:
str,
cus-
tom_emoji
str
|
None
=
None,
**ex-
```

tra\_data:
Any)

Use this method to set the thumbnail of a custom emoji sticker set. Returns True on success.

Source: https://core.telegram.org/bots/api#setcustomemojistickersetthumbnail

name: str

Sticker set name

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model post init method.

#### custom\_emoji\_id: str | None

Custom emoji identifier of a sticker from the sticker set; pass an empty string to drop the thumbnail and use the first sticker as the thumbnail.

## **Usage**

## As bot method

```
result: bool = await bot.set_custom_emoji_sticker_set_thumbnail(...)
```

## Method as object

Imports:

- from aiogram.methods.set\_custom\_emoji\_sticker\_set\_thumbnail import SetCustomEmojiStickerSetThumbnail
- alias: from aiogram.methods import SetCustomEmojiStickerSetThumbnail

# With specific bot

```
result: bool = await bot(SetCustomEmojiStickerSetThumbnail(...))
```

## As reply into Webhook in handler

```
return SetCustomEmojiStickerSetThumbnail(...)
```

## setStickerEmojiList

Returns: bool

Use this method to change the list of emoji assigned to a regular or custom emoji sticker. The sticker must belong to a sticker set created by the bot. Returns True on success.

Source: https://core.telegram.org/bots/api#setstickeremojilist

sticker: str

File identifier of the sticker

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
emoji_list: List[str]
```

A JSON-serialized list of 1-20 emoji associated with the sticker

#### As bot method

```
result: bool = await bot.set_sticker_emoji_list(...)
```

## Method as object

Imports:

- from aiogram.methods.set\_sticker\_emoji\_list import SetStickerEmojiList
- alias: from aiogram.methods import SetStickerEmojiList

## With specific bot

```
result: bool = await bot(SetStickerEmojiList(...))
```

# As reply into Webhook in handler

```
return SetStickerEmojiList(...)
```

## setStickerKeywords

Returns: bool

Use this method to change search keywords assigned to a regular or custom emoji sticker. The sticker must belong to a sticker set created by the bot. Returns True on success.

Source: https://core.telegram.org/bots/api#setstickerkeywords

sticker: str

File identifier of the sticker

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

keywords: List[str] | None

A JSON-serialized list of 0-20 search keywords for the sticker with total length of up to 64 characters

### As bot method

```
result: bool = await bot.set_sticker_keywords(...)
```

## Method as object

Imports:

- from aiogram.methods.set\_sticker\_keywords import SetStickerKeywords
- alias: from aiogram.methods import SetStickerKeywords

## With specific bot

```
result: bool = await bot(SetStickerKeywords(...))
```

# As reply into Webhook in handler

```
return SetStickerKeywords(...)
```

## setStickerMaskPosition

Returns: bool

Use this method to change the mask position of a mask sticker. The sticker must belong to a sticker set that was created by the bot. Returns True on success.

Source: https://core.telegram.org/bots/api#setstickermaskposition

```
sticker: str
```

File identifier of the sticker

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
mask_position: MaskPosition | None
```

A JSON-serialized object with the position where the mask should be placed on faces. Omit the parameter to remove the mask position.

#### As bot method

```
result: bool = await bot.set_sticker_mask_position(...)
```

## Method as object

Imports:

- from aiogram.methods.set\_sticker\_mask\_position import SetStickerMaskPosition
- alias: from aiogram.methods import SetStickerMaskPosition

## With specific bot

```
result: bool = await bot(SetStickerMaskPosition(...))
```

# As reply into Webhook in handler

```
return SetStickerMaskPosition(...)
```

## setStickerPositionInSet

```
Returns: bool
```

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

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

```
sticker: str
```

File identifier of the sticker

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

position: int

New sticker position in the set, zero-based

#### As bot method

```
result: bool = await bot.set_sticker_position_in_set(...)
```

## Method as object

Imports:

- from aiogram.methods.set\_sticker\_position\_in\_set import SetStickerPositionInSet
- alias: from aiogram.methods import SetStickerPositionInSet

## With specific bot

```
result: bool = await bot(SetStickerPositionInSet(...))
```

# As reply into Webhook in handler

```
return SetStickerPositionInSet(...)
```

## As shortcut from received object

• aiogram.types.sticker.Sticker.set\_position\_in\_set()

#### setStickerSetThumbnail

Returns: bool

Use this method to set the thumbnail of a regular or mask sticker set. The format of the thumbnail file must match the format of the stickers in the set. Returns True on success.

Source: https://core.telegram.org/bots/api#setstickersetthumbnail

name: str

Sticker set name

user\_id: int

User identifier of the sticker set owner

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### format: str

Format of the thumbnail, must be one of 'static' for a .WEBP or .PNG image, 'animated' for a .TGS animation, or 'video' for a WEBM video

```
thumbnail: InputFile | str | None
```

A .WEBP or .PNG image with the thumbnail, must be up to 128 kilobytes in size and have a width and height of exactly 100px, or a .TGS animation with a thumbnail up to 32 kilobytes in size (see https://core.telegram.org/stickers#animation-requirements <a href="https://core.telegram.org/stickers#animation-requirements">https://core.telegram.org/stickers#animation-requirements</a> for animated sticker technical requirements), or a WEBM video with the thumbnail up to 32 kilobytes in size; see https://core.telegram.org/stickers#video-requirements <a href="https://core.telegram.org/stickers#video-requirements">https://core.telegram.org/stickers#video-requirements</a> <a href="https://core.telegram.org/stickers#video-requirements">https://core.telegram.org/stickers#video-requirements</a> <a href="https://core.telegram.org/stickers#video-requirements">https://core.telegram.org/stickers#video-requirements</a> <a href="https://core.telegram.org/stickers#video-requirements">https://core.telegram.org/stickers#video-requirements</a> for video sticker technical requirements. Pass a file\_id as a String to send a file that already exists on the Telegram servers, pass an HTTP URL as a String for Telegram to get a file from the Internet, or upload a new one using multipart/form-data. \*More information on Sending Files\*\*. Animated and video sticker set thumbnails can't be uploaded via HTTP URL. If omitted, then the thumbnail is dropped and the first sticker is used as the thumbnail.

## **Usage**

#### As bot method

```
result: bool = await bot.set_sticker_set_thumbnail(...)
```

# Method as object

Imports:

- $\bullet \ \ from \ aiogram.methods.set\_sticker\_set\_thumbnail \ import \ SetStickerSetThumbnail$
- alias: from aiogram.methods import SetStickerSetThumbnail

## With specific bot

```
result: bool = await bot(SetStickerSetThumbnail(...))
```

#### As reply into Webhook in handler

```
return SetStickerSetThumbnail(...)
```

## setStickerSetTitle

```
Returns: bool
```

Use this method to set the title of a created sticker set. Returns True on success.

Source: https://core.telegram.org/bots/api#setstickersettitle

name: str

Sticker set name

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

title: str

Sticker set title, 1-64 characters

## **Usage**

#### As bot method

```
result: bool = await bot.set_sticker_set_title(...)
```

## Method as object

Imports:

- from aiogram.methods.set\_sticker\_set\_title import SetStickerSetTitle
- alias: from aiogram.methods import SetStickerSetTitle

## With specific bot

```
result: bool = await bot(SetStickerSetTitle(...))
```

## As reply into Webhook in handler

```
return SetStickerSetTitle(...)
```

## uploadStickerFile

Returns: File

Use this method to upload a file with a sticker for later use in the aiogram.methods. create\_new\_sticker\_set.CreateNewStickerSet, aiogram.methods.add\_sticker\_to\_set. AddStickerToSet, or aiogram.methods.replace\_sticker\_in\_set.ReplaceStickerInSet methods (the file can be used multiple times). Returns the uploaded aiogram.types.file.File on success.

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

## user\_id: int

User identifier of sticker file owner

## sticker: InputFile

A file with the sticker in .WEBP, .PNG, .TGS, or .WEBM format. See https://core.telegram.org/stickers <a href="https://core.telegram.org/stickers">https://core.telegram.org/stickers</a> information on Sending Files »

## model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### sticker\_format: str

Format of the sticker, must be one of 'static', 'animated', 'video'

## **Usage**

#### As bot method

```
result: File = await bot.upload_sticker_file(...)
```

## Method as object

#### Imports:

- from aiogram.methods.upload\_sticker\_file import UploadStickerFile
- alias: from aiogram.methods import UploadStickerFile

```
result: File = await bot(UploadStickerFile(...))
```

#### Available methods

## answerCallbackQuery

Returns: bool

```
class aiogram.methods.answer_callback_query.AnswerCallbackQuery(*, callback_query_id: str, text: str \mid None = None, show\_alert: bool \mid None = None, url: str \mid None = None, cache\_time: int \mid None = None, **extra_data: Any)
```

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. On success, True is returned.

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

## callback\_query\_id: str

Unique identifier for the query to be answered

## text: str | None

Text of the notification. If not specified, nothing will be shown to the user, 0-200 characters

#### show\_alert: bool | None

If True, an alert will be shown by the client instead of a notification at the top of the chat screen. Defaults to *false*.

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
url: str | None
```

URL that will be opened by the user's client. If you have created a <code>aiogram.types.game.Game</code> and accepted the conditions via <code>@BotFather</code>, specify the URL that opens your game - note that this will only work if the query comes from a <code>https://core.telegram.org/bots/api#inlinekeyboardbutton</code> <code>callback\_game</code> button.

```
cache_time: int | None
```

The maximum amount of time in seconds that the result of the callback query may be cached client-side. Telegram apps will support caching starting in version 3.14. Defaults to 0.

#### As bot method

```
result: bool = await bot.answer_callback_query(...)
```

# Method as object

Imports:

- from aiogram.methods.answer\_callback\_query import AnswerCallbackQuery
- alias: from aiogram.methods import AnswerCallbackQuery

## With specific bot

```
result: bool = await bot(AnswerCallbackQuery(...))
```

## As reply into Webhook in handler

```
return AnswerCallbackQuery(...)
```

#### As shortcut from received object

• aiogram.types.callback\_query.CallbackQuery.answer()

# approveChatJoinRequest

Returns: bool

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

Source: https://core.telegram.org/bots/api#approvechatjoinrequest

```
chat_id: int | str
```

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

user\_id: int

Unique identifier of the target user

#### As bot method

```
result: bool = await bot.approve_chat_join_request(...)
```

## Method as object

Imports:

- from aiogram.methods.approve\_chat\_join\_request import ApproveChatJoinRequest
- alias: from aiogram.methods import ApproveChatJoinRequest

## With specific bot

```
result: bool = await bot(ApproveChatJoinRequest(...))
```

# As reply into Webhook in handler

```
return ApproveChatJoinRequest(...)
```

## As shortcut from received object

• aiogram.types.chat\_join\_request.ChatJoinRequest.approve()

#### banChatMember

Returns: bool

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

Source: https://core.telegram.org/bots/api#banchatmember

```
chat_id: int | str
```

Unique identifier for the target group or username of the target supergroup or channel (in the format @channelusername)

```
user_id: int
```

Unique identifier of the target user

## model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
until_date: datetime.datetime | datetime.timedelta | int | None
```

Date when the user will be unbanned; Unix time. If user is banned for more than 366 days or less than 30 seconds from the current time they are considered to be banned forever. Applied for supergroups and channels only.

```
revoke_messages: bool | None
```

Pass True to delete all messages from the chat for the user that is being removed. If False, the user will be able to see messages in the group that were sent before the user was removed. Always True for supergroups and channels.

# **Usage**

#### As bot method

```
result: bool = await bot.ban_chat_member(...)
```

## Method as object

## Imports:

- from aiogram.methods.ban\_chat\_member import BanChatMember
- alias: from aiogram.methods import BanChatMember

# With specific bot

```
result: bool = await bot(BanChatMember(...))
```

## As reply into Webhook in handler

```
return BanChatMember(...)
```

## As shortcut from received object

• aiogram.types.chat.Chat.ban()

## banChatSenderChat

Returns: bool

Use this method to ban a channel chat in a supergroup or a channel. Until the chat is unbanned, the owner of the banned chat won't be able to send messages on behalf of **any of their channels**. The bot must be an administrator in the supergroup or channel for this to work and must have the appropriate administrator rights. Returns True on success.

Source: https://core.telegram.org/bots/api#banchatsenderchat

```
chat_id: int | str
```

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### sender\_chat\_id: int

Unique identifier of the target sender chat

## **Usage**

# As bot method

```
result: bool = await bot.ban_chat_sender_chat(...)
```

## Method as object

Imports:

- from aiogram.methods.ban\_chat\_sender\_chat import BanChatSenderChat
- alias: from aiogram.methods import BanChatSenderChat

## With specific bot

```
result: bool = await bot(BanChatSenderChat(...))
```

## As reply into Webhook in handler

```
return BanChatSenderChat(...)
```

# As shortcut from received object

• aiogram.types.chat.Chat.ban\_sender\_chat()

#### close

Returns: bool

```
class aiogram.methods.close.Close(**extra_data: Any)
```

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

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

## **Usage**

## As bot method

```
result: bool = await bot.close(...)
```

# Method as object

# Imports:

- from aiogram.methods.close import Close
- alias: from aiogram.methods import Close

```
result: bool = await bot(Close(...))
```

# As reply into Webhook in handler

```
return Close(...)
```

## closeForumTopic

Returns: bool

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

Source: https://core.telegram.org/bots/api#closeforumtopic

```
chat_id: int | str
```

Unique identifier for the target chat or username of the target supergroup (in the format @supergroupusername)

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
message_thread_id: int
```

Unique identifier for the target message thread of the forum topic

## **Usage**

#### As bot method

```
result: bool = await bot.close_forum_topic(...)
```

## Method as object

Imports:

- from aiogram.methods.close\_forum\_topic import CloseForumTopic
- alias: from aiogram.methods import CloseForumTopic

# With specific bot

```
result: bool = await bot(CloseForumTopic(...))
```

# As reply into Webhook in handler

```
return CloseForumTopic(...)
```

# closeGeneralForumTopic

Returns: bool

Use this method to close an open 'General' topic in a forum supergroup chat. The bot must be an administrator in the chat for this to work and must have the *can manage topics* administrator rights. Returns True on success.

Source: https://core.telegram.org/bots/api#closegeneralforumtopic

```
chat_id: int | str
```

Unique identifier for the target chat or username of the target supergroup (in the format @supergroupusername)

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

## **Usage**

# As bot method

```
result: bool = await bot.close_general_forum_topic(...)
```

# Method as object

Imports:

- from aiogram.methods.close\_general\_forum\_topic import CloseGeneralForumTopic
- alias: from aiogram.methods import CloseGeneralForumTopic

## With specific bot

```
result: bool = await bot(CloseGeneralForumTopic(...))
```

# As reply into Webhook in handler

```
return CloseGeneralForumTopic(...)
```

## copyMessage

Returns: MessageId

```
class aiogram.methods.copy_message.CopyMessage(*, chat_id: int | str, from_chat_id: int | str,
                                                          message_id: int, message_thread_id: int | None =
                                                          None, caption: str \mid None = None, parse mode: str \mid
                                                           \simaiogram.client.default.Default | None =
                                                           <Default('parse mode')>, caption entities: ~typ-
                                                           ing.List[~aiogram.types.message_entity.MessageEntity]
                                                           | None = None, show caption above media: bool |
                                                           ~aiogram.client.default.Default | None =
                                                           <Default('show_caption_above_media')>,
                                                           disable\_notification: bool \mid None = None,
                                                          protect_content: bool | ~aiogram.client.default.Default
                                                          | None = < Default('protect_content')>,
                                                           reply_parameters:
                                                           ~aiogram.types.reply_parameters.ReplyParameters |
                                                          None = None, reply markup:
                                                           ~aiogram.types.inline_keyboard_markup.InlineKeyboardMarkup
                                                           ~aiogram.types.reply_keyboard_markup.ReplyKeyboardMarkup
                                                           ~aiogram.types.reply_keyboard_remove.ReplyKeyboardRemove
                                                           | ~aiogram.types.force_reply.ForceReply | None =
                                                          None, allow sending without reply: bool | None =
                                                          None, reply\_to\_message\_id: int | None = None,
                                                           **extra_data: ~typing.Any)
```

Use this method to copy messages of any kind. Service messages, paid media messages, giveaway messages, giveaway winners messages, and invoice messages can't be copied. A quiz aiogram.methods.poll.Poll can be copied only if the value of the field <code>correct\_option\_id</code> is known to the bot. The method is analogous to the method <code>aiogram.methods.forward\_message.ForwardMessage</code>, but the copied message doesn't have a link to the original message. Returns the <code>aiogram.types.message\_id.MessageId</code> of the sent message on success.

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

#### chat\_id: int | str

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

# from\_chat\_id: int | str

Unique identifier for the chat where the original message was sent (or channel username in the format @channelusername)

#### message\_id: int

Message identifier in the chat specified in from\_chat\_id

## message\_thread\_id: int | None

Unique identifier for the target message thread (topic) of the forum; for forum supergroups only

#### caption: str | None

New caption for media, 0-1024 characters after entities parsing. If not specified, the original caption is kept

#### parse\_mode: str | Default | None

Mode for parsing entities in the new caption. See formatting options for more details.

#### caption\_entities: List[MessageEntity] | None

A JSON-serialized list of special entities that appear in the new caption, which can be specified instead of *parse\_mode* 

# model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

# show\_caption\_above\_media: bool | Default | None

Pass True, if the caption must be shown above the message media. Ignored if a new caption isn't specified.

#### disable notification: bool | None

Sends the message silently. Users will receive a notification with no sound.

## protect\_content: bool | Default | None

Protects the contents of the sent message from forwarding and saving

## reply\_parameters: ReplyParameters | None

Description of the message to reply to

# reply\_markup: InlineKeyboardMarkup | ReplyKeyboardMarkup | ReplyKeyboardRemove | ForceReply | None

Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

## allow\_sending\_without\_reply: bool | None

Pass True if the message should be sent even if the specified replied-to message is not found

Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023

#### reply\_to\_message\_id: int | None

If the message is a reply, ID of the original message

Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023

#### As bot method

```
result: MessageId = await bot.copy_message(...)
```

## Method as object

Imports:

- from aiogram.methods.copy\_message import CopyMessage
- alias: from aiogram.methods import CopyMessage

## With specific bot

```
result: MessageId = await bot(CopyMessage(...))
```

# As reply into Webhook in handler

```
return CopyMessage(...)
```

## As shortcut from received object

• aiogram.types.message.Message.copy\_to()

#### copyMessages

Returns: List[MessageId]

```
class aiogram.methods.copy_messages.CopyMessages(*, chat_id: int | str, from_chat_id: int | str, message_ids: List[int], message_thread_id: int | None = None, disable_notification: bool | None = None, protect_content: bool | None = None, remove_caption: bool | None = None, **extra_data: Any)
```

Use this method to copy messages of any kind. If some of the specified messages can't be found or copied, they are skipped. Service messages, paid media messages, giveaway messages, giveaway winners messages, and invoice messages can't be copied. A quiz aiogram.methods.poll.Poll can be copied only if the value of the field <code>correct\_option\_id</code> is known to the bot. The method is analogous to the method <code>aiogram.methods.forward\_messages.ForwardMessages</code>, but the copied messages don't have a link to the original message. Album grouping is kept for copied messages. On success, an array of <code>aiogram.types.message\_id.MessageId</code> of the sent messages is returned.

Source: https://core.telegram.org/bots/api#copymessages

#### chat\_id: int | str

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

#### from\_chat\_id: int | str

Unique identifier for the chat where the original messages were sent (or channel username in the format @channelusername)

## message\_ids: List[int]

A JSON-serialized list of 1-100 identifiers of messages in the chat *from\_chat\_id* to copy. The identifiers must be specified in a strictly increasing order.

#### message\_thread\_id: int | None

Unique identifier for the target message thread (topic) of the forum; for forum supergroups only

#### model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

# disable\_notification: bool | None

Sends the messages silently. Users will receive a notification with no sound.

```
protect_content: bool | None
```

Protects the contents of the sent messages from forwarding and saving

#### remove\_caption: bool | None

Pass True to copy the messages without their captions

#### **Usage**

#### As bot method

```
result: List[MessageId] = await bot.copy_messages(...)
```

## Method as object

## Imports:

- from aiogram.methods.copy\_messages import CopyMessages
- alias: from aiogram.methods import CopyMessages

# With specific bot

```
result: List[MessageId] = await bot(CopyMessages(...))
```

## As reply into Webhook in handler

```
return CopyMessages(...)
```

#### createChatInviteLink

Returns: ChatInviteLink

class aiogram.methods.create\_chat\_invite\_link.CreateChatInviteLink(\*, chat\_id: int | str, name:

str | None = None, expire\_date: datetime | timedelta | int | None = None, member\_limit: int | None = None, creates\_join\_request: bool | None = None, \*\*extra\_data: Any)

Use this method to create an additional invite link for a chat. The bot must be an administrator in the chat for this to work and must have the appropriate administrator rights. The link can be revoked using the method <code>aiogram.methods.revoke\_chat\_invite\_link.RevokeChatInviteLink</code>. Returns the new invite link as <code>aiogram.types.chat\_invite\_link.ChatInviteLink</code> object.

Source: https://core.telegram.org/bots/api#createchatinvitelink

#### chat\_id: int | str

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

#### name: str | None

Invite link name; 0-32 characters

#### expire\_date: datetime.datetime | datetime.timedelta | int | None

Point in time (Unix timestamp) when the link will expire

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
member_limit: int | None
```

The maximum number of users that can be members of the chat simultaneously after joining the chat via this invite link; 1-99999

```
creates_join_request: bool | None
```

True, if users joining the chat via the link need to be approved by chat administrators. If True, *member\_limit* can't be specified

#### As bot method

```
result: ChatInviteLink = await bot.create_chat_invite_link(...)
```

# Method as object

## Imports:

- from aiogram.methods.create\_chat\_invite\_link import CreateChatInviteLink
- alias: from aiogram.methods import CreateChatInviteLink

## With specific bot

```
result: ChatInviteLink = await bot(CreateChatInviteLink(...))
```

# As reply into Webhook in handler

```
return CreateChatInviteLink(...)
```

# As shortcut from received object

• aiogram.types.chat.Chat.create\_invite\_link()

## createChatSubscriptionInviteLink

Returns: ChatInviteLink

class aiogram.methods.create\_chat\_subscription\_invite\_link.CreateChatSubscriptionInviteLink(\*,

```
chat_id:
int
str.
sub-
scrip-
tion period:
date-
time
timedelta
int,
sub-
scrip-
tion_price:
int.
name:
str
None
None,
**ex-
tra_data:
Any)
```

Use this method to create a subscription invite link for a channel chat. The bot must have the  $can\_invite\_users$  administrator rights. The link can be edited using the method aiogram.methods.  $edit\_chat\_subscription\_invite\_link.EditChatSubscriptionInviteLink$  or revoked using the method  $aiogram.methods.revoke\_chat\_invite\_link.RevokeChatInviteLink$ . Returns the new invite link as a  $aiogram.types.chat\_invite\_link.ChatInviteLink$  object.

Source: https://core.telegram.org/bots/api#createchatsubscriptioninvitelink

#### chat\_id: int | str

Unique identifier for the target channel chat or username of the target channel (in the format @channelusername)

#### subscription\_period: datetime.datetime | datetime.timedelta | int

The number of seconds the subscription will be active for before the next payment. Currently, it must always be 2592000 (30 days).

#### model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

#### $model_post_init(context: Any, /) \rightarrow None$

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### subscription\_price: int

The amount of Telegram Stars a user must pay initially and after each subsequent subscription period to be a member of the chat: 1-2500

```
name: str | None
```

Invite link name; 0-32 characters

## **Usage**

#### As bot method

```
result: ChatInviteLink = await bot.create_chat_subscription_invite_link(...)
```

## Method as object

Imports:

- from aiogram.methods.create\_chat\_subscription\_invite\_link import CreateChatSubscriptionInviteLink
- alias: from aiogram.methods import CreateChatSubscriptionInviteLink

# With specific bot

```
result: ChatInviteLink = await bot(CreateChatSubscriptionInviteLink(...))
```

## As reply into Webhook in handler

```
return CreateChatSubscriptionInviteLink(...)
```

#### createForumTopic

Returns: ForumTopic

Use this method to create a topic in a forum supergroup chat. The bot must be an administrator in the chat for this to work and must have the *can\_manage\_topics* administrator rights. Returns information about the created topic as a *aiogram.types.forum\_topic.ForumTopic* object.

Source: https://core.telegram.org/bots/api#createforumtopic

```
chat_id: int | str
```

Unique identifier for the target chat or username of the target supergroup (in the format @supergroupusername)

name: str

Topic name, 1-128 characters

## model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
icon_color: int | None
```

Color of the topic icon in RGB format. Currently, must be one of 7322096 (0x6FB9F0), 16766590 (0xFFD67E), 13338331 (0xCB86DB), 9367192 (0x8EEE98), 16749490 (0xFF93B2), or 16478047 (0xFB6F5F)

```
icon_custom_emoji_id: str | None
```

Unique identifier of the custom emoji shown as the topic icon. Use <code>aiogram.methods.get\_forum\_topic\_icon\_stickers.GetForumTopicIconStickers</code> to get all allowed custom emoji identifiers.

## **Usage**

#### As bot method

```
result: ForumTopic = await bot.create_forum_topic(...)
```

# Method as object

Imports:

- from aiogram.methods.create\_forum\_topic import CreateForumTopic
- alias: from aiogram.methods import CreateForumTopic

# With specific bot

```
result: ForumTopic = await bot(CreateForumTopic(...))
```

# As reply into Webhook in handler

```
return CreateForumTopic(...)
```

#### declineChatJoinRequest

Returns: bool

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

Source: https://core.telegram.org/bots/api#declinechatjoinrequest

# chat\_id: int | str

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

# model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### user\_id: int

Unique identifier of the target user

#### **Usage**

#### As bot method

```
result: bool = await bot.decline_chat_join_request(...)
```

# Method as object

#### Imports:

- from aiogram.methods.decline\_chat\_join\_request import DeclineChatJoinRequest
- alias: from aiogram.methods import DeclineChatJoinRequest

# With specific bot

```
result: bool = await bot(DeclineChatJoinRequest(...))
```

#### As reply into Webhook in handler

```
return DeclineChatJoinRequest(...)
```

# As shortcut from received object

• aiogram.types.chat\_join\_request.ChatJoinRequest.decline()

#### deleteChatPhoto

Returns: bool

class aiogram.methods.delete\_chat\_photo.DeleteChatPhoto(\*, chat\_id: int | str, \*\*extra\_data: Any)

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 administrator rights. Returns True on success.

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

```
chat_id: int | str
```

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

# **Usage**

#### As bot method

```
result: bool = await bot.delete_chat_photo(...)
```

## Method as object

Imports:

- from aiogram.methods.delete\_chat\_photo import DeleteChatPhoto
- alias: from aiogram.methods import DeleteChatPhoto

#### With specific bot

```
result: bool = await bot(DeleteChatPhoto(...))
```

## As reply into Webhook in handler

```
return DeleteChatPhoto(...)
```

## As shortcut from received object

• aiogram.types.chat.Chat.delete\_photo()

#### deleteChatStickerSet

Returns: bool

Use this method to delete a group sticker set from a supergroup. The bot must be an administrator in the chat for this to work and must have the appropriate administrator rights. Use the field <code>can\_set\_sticker\_set</code> optionally returned in <code>aiogram.methods.get\_chat.GetChat</code> requests to check if the bot can use this method. Returns True on success.

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

```
chat_id: int | str
```

Unique identifier for the target chat or username of the target supergroup (in the format @supergroupusername)

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### **Usage**

## As bot method

```
result: bool = await bot.delete_chat_sticker_set(...)
```

# Method as object

Imports:

- from aiogram.methods.delete\_chat\_sticker\_set import DeleteChatStickerSet
- alias: from aiogram.methods import DeleteChatStickerSet

#### With specific bot

```
result: bool = await bot(DeleteChatStickerSet(...))
```

## As reply into Webhook in handler

```
return DeleteChatStickerSet(...)
```

# As shortcut from received object

• aiogram.types.chat.Chat.delete\_sticker\_set()

# deleteForumTopic

Returns: bool

Use this method to delete a forum topic along with all its messages in a forum supergroup chat. The bot must be an administrator in the chat for this to work and must have the *can\_delete\_messages* administrator rights. Returns True on success.

Source: https://core.telegram.org/bots/api#deleteforumtopic

```
chat_id: int | str
```

Unique identifier for the target chat or username of the target supergroup (in the format @supergroupusername)

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
message_thread_id: int
```

Unique identifier for the target message thread of the forum topic

#### **Usage**

#### As bot method

```
result: bool = await bot.delete_forum_topic(...)
```

## Method as object

Imports:

- from aiogram.methods.delete\_forum\_topic import DeleteForumTopic
- alias: from aiogram.methods import DeleteForumTopic

## With specific bot

```
result: bool = await bot(DeleteForumTopic(...))
```

# As reply into Webhook in handler

```
return DeleteForumTopic(...)
```

## deleteMyCommands

Returns: bool

class aiogram.methods.delete\_my\_commands.DeleteMyCommands(\*, scope: BotCommandScopeAllPrivateChats |
BotCommandScopeAllGroupChats |
BotCommandScopeAllChatAdministrators | BotCommandScopeChat |
BotCommandScopeChatAdministrators |
BotCommandScopeChatMember | None = None, language\_code: str | None = None, \*\*extra\_data: Any)

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

Source: https://core.telegram.org/bots/api#deletemycommands

```
scope: BotCommandScopeDefault | BotCommandScopeAllPrivateChats |
BotCommandScopeAllGroupChats | BotCommandScopeAllChatAdministrators |
BotCommandScopeChat | BotCommandScopeChatAdministrators | BotCommandScopeChatMember |
None
```

A JSON-serialized object, describing scope of users for which the commands are relevant. Defaults to aiogram.types.bot\_command\_scope\_default.BotCommandScopeDefault.

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
language_code: str | None
```

A two-letter ISO 639-1 language code. If empty, commands will be applied to all users from the given scope, for whose language there are no dedicated commands

#### As bot method

```
result: bool = await bot.delete_my_commands(...)
```

## Method as object

Imports:

- from aiogram.methods.delete\_my\_commands import DeleteMyCommands
- alias: from aiogram.methods import DeleteMyCommands

## With specific bot

```
result: bool = await bot(DeleteMyCommands(...))
```

# As reply into Webhook in handler

```
return DeleteMyCommands(...)
```

#### editChatInviteLink

Returns: ChatInviteLink

str, name: str | None = None, expire\_date: datetime | timedelta | int | None = None, member\_limit: int | None = None, creates\_join\_request: bool | None = None, \*\*extra\_data: Any)

Use this method to edit a non-primary invite link created by the bot. The bot must be an administrator in the chat for this to work and must have the appropriate administrator rights. Returns the edited invite link as a <code>aiogram.types.chat\_invite\_link.ChatInviteLink</code> object.

Source: https://core.telegram.org/bots/api#editchatinvitelink

```
chat_id: int | str
```

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

invite\_link: str

The invite link to edit

name: str | None

Invite link name; 0-32 characters

## model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
expire_date: datetime.datetime | datetime.timedelta | int | None
```

Point in time (Unix timestamp) when the link will expire

```
member_limit: int | None
```

The maximum number of users that can be members of the chat simultaneously after joining the chat via this invite link; 1-99999

```
creates_join_request: bool | None
```

True, if users joining the chat via the link need to be approved by chat administrators. If True, *member\_limit* can't be specified

# **Usage**

#### As bot method

```
result: ChatInviteLink = await bot.edit_chat_invite_link(...)
```

# Method as object

## Imports:

- from aiogram.methods.edit\_chat\_invite\_link import EditChatInviteLink
- alias: from aiogram.methods import EditChatInviteLink

#### With specific bot

```
result: ChatInviteLink = await bot(EditChatInviteLink(...))
```

# As reply into Webhook in handler

```
return EditChatInviteLink(...)
```

#### As shortcut from received object

• aiogram.types.chat.Chat.edit\_invite\_link()

# editChatSubscriptionInviteLink

Returns: ChatInviteLink

class aiogram.methods.edit\_chat\_subscription\_invite\_link.EditChatSubscriptionInviteLink(\*,

```
chat_id:
int
|
str,
in-
vite_link:
str,
name:
str
|
None
=
None,
**ex-
tra_data:
Any)
```

Use this method to edit a subscription invite link created by the bot. The bot must have the *can\_invite\_users* administrator rights. Returns the edited invite link as a *aiogram.types.chat\_invite\_link*. ChatInviteLink object.

Source: https://core.telegram.org/bots/api#editchatsubscriptioninvitelink

```
chat_id: int | str
```

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

#### invite\_link: str

The invite link to edit

# model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
name: str | None
```

Invite link name; 0-32 characters

#### **Usage**

#### As bot method

```
result: ChatInviteLink = await bot.edit_chat_subscription_invite_link(...)
```

## Method as object

Imports:

- from aiogram.methods.edit\_chat\_subscription\_invite\_link import EditChatSubscriptionInviteLink
- alias: from aiogram.methods import EditChatSubscriptionInviteLink

# With specific bot

```
result: ChatInviteLink = await bot(EditChatSubscriptionInviteLink(...))
```

## As reply into Webhook in handler

```
return EditChatSubscriptionInviteLink(...)
```

## editForumTopic

Returns: bool

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

Source: https://core.telegram.org/bots/api#editforumtopic

```
chat_id: int | str
```

Unique identifier for the target chat or username of the target supergroup (in the format @supergroupusername)

```
message_thread_id: int
```

Unique identifier for the target message thread of the forum topic

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
name: str | None
```

New topic name, 0-128 characters. If not specified or empty, the current name of the topic will be kept

```
icon_custom_emoji_id: str | None
```

New unique identifier of the custom emoji shown as the topic icon. Use <code>aiogram.methods.get\_forum\_topic\_icon\_stickers.GetForumTopicIconStickers</code> to get all allowed custom emoji identifiers. Pass an empty string to remove the icon. If not specified, the current icon will be kept

#### As bot method

```
result: bool = await bot.edit_forum_topic(...)
```

## Method as object

Imports:

- from aiogram.methods.edit\_forum\_topic import EditForumTopic
- alias: from aiogram.methods import EditForumTopic

## With specific bot

```
result: bool = await bot(EditForumTopic(...))
```

# As reply into Webhook in handler

```
return EditForumTopic(...)
```

## editGeneralForumTopic

Returns: bool

Use this method to edit the name of the 'General' topic in a forum supergroup chat. The bot must be an administrator in the chat for this to work and must have the *can\_manage\_topics* administrator rights. Returns True on success.

Source: https://core.telegram.org/bots/api#editgeneralforumtopic

```
chat_id: int | str
```

Unique identifier for the target chat or username of the target supergroup (in the format @supergroupusername)

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

name: str

New topic name, 1-128 characters

#### As bot method

```
result: bool = await bot.edit_general_forum_topic(...)
```

## Method as object

Imports:

- from aiogram.methods.edit\_general\_forum\_topic import EditGeneralForumTopic
- alias: from aiogram.methods import EditGeneralForumTopic

## With specific bot

```
result: bool = await bot(EditGeneralForumTopic(...))
```

# As reply into Webhook in handler

```
return EditGeneralForumTopic(...)
```

## exportChatInviteLink

Returns: str

Use this method to generate a new primary invite link for a chat; any previously generated primary link is revoked. The bot must be an administrator in the chat for this to work and must have the appropriate administrator rights. Returns the new invite link as *String* on success.

Note: Each administrator in a chat generates their own invite links. Bots can't use invite links generated by other administrators. If you want your bot to work with invite links, it will need to generate its own link using <code>aiogram.methods.export\_chat\_invite\_link.ExportChatInviteLink</code> or by calling the <code>aiogram.methods.get\_chat.GetChat</code> method. If your bot needs to generate a new primary invite link replacing its previous one, use <code>aiogram.methods.export\_chat\_invite\_link.ExportChatInvite\_link</code> again.

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

```
chat_id: int | str
```

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### As bot method

```
result: str = await bot.export_chat_invite_link(...)
```

# Method as object

Imports:

- from aiogram.methods.export\_chat\_invite\_link import ExportChatInviteLink
- alias: from aiogram.methods import ExportChatInviteLink

# With specific bot

```
result: str = await bot(ExportChatInviteLink(...))
```

# As reply into Webhook in handler

```
        return
        ExportChatInviteLink(...)
```

#### As shortcut from received object

• aiogram.types.chat.Chat.export\_invite\_link()

## forwardMessage

Returns: Message

Use this method to forward messages of any kind. Service messages and messages with protected content can't be forwarded. On success, the sent <code>aiogram.types.message.Message</code> is returned.

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

```
chat_id: int | str
```

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

```
from_chat_id: int | str
```

Unique identifier for the chat where the original message was sent (or channel username in the format @channelusername)

## message\_id: int

Message identifier in the chat specified in from\_chat\_id

# model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
message_thread_id: int | None
```

Unique identifier for the target message thread (topic) of the forum; for forum supergroups only

```
disable_notification: bool | None
```

Sends the message silently. Users will receive a notification with no sound.

```
protect_content: bool | Default | None
```

Protects the contents of the forwarded message from forwarding and saving

## **Usage**

#### As bot method

```
result: Message = await bot.forward_message(...)
```

## Method as object

#### Imports:

- from aiogram.methods.forward\_message import ForwardMessage
- alias: from aiogram.methods import ForwardMessage

#### With specific bot

```
result: Message = await bot(ForwardMessage(...))
```

# As reply into Webhook in handler

```
return ForwardMessage(...)
```

## As shortcut from received object

• aiogram.types.message.Message.forward()

# forwardMessages

Returns: List[MessageId]

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

Source: https://core.telegram.org/bots/api#forwardmessages

```
chat_id: int | str
```

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

```
from_chat_id: int | str
```

Unique identifier for the chat where the original messages were sent (or channel username in the format @channelusername)

```
message_ids: List[int]
```

A JSON-serialized list of 1-100 identifiers of messages in the chat *from\_chat\_id* to forward. The identifiers must be specified in a strictly increasing order.

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
message_thread_id: int | None
```

Unique identifier for the target message thread (topic) of the forum; for forum supergroups only

```
disable_notification: bool | None
```

Sends the messages silently. Users will receive a notification with no sound.

```
protect_content: bool | None
```

Protects the contents of the forwarded messages from forwarding and saving

#### As bot method

```
result: List[MessageId] = await bot.forward_messages(...)
```

## Method as object

Imports:

- from aiogram.methods.forward\_messages import ForwardMessages
- alias: from aiogram.methods import ForwardMessages

## With specific bot

```
result: List[MessageId] = await bot(ForwardMessages(...))
```

# As reply into Webhook in handler

```
return ForwardMessages(...)
```

## getBusinessConnection

Returns: BusinessConnection

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

Source: https://core.telegram.org/bots/api#getbusinessconnection

business\_connection\_id: str

Unique identifier of the business connection

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### As bot method

```
result: BusinessConnection = await bot.get_business_connection(...)
```

## Method as object

Imports:

- $\bullet \ \ from \ a iogram.methods.get\_business\_connection \ import \ GetBusinessConnection$
- alias: from aiogram.methods import GetBusinessConnection

## With specific bot

```
result: BusinessConnection = await bot(GetBusinessConnection(...))
```

# getChat

Returns: ChatFullInfo

```
class aiogram.methods.get_chat.GetChat(*, chat_id: int | str, **extra_data: Any)
```

Use this method to get up-to-date information about the chat. Returns a aiogram.types.chat\_full\_info. ChatFullInfo object on success.

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

```
chat_id: int | str
```

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

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

# **Usage**

#### As bot method

```
result: ChatFullInfo = await bot.get_chat(...)
```

# Method as object

#### Imports:

- from aiogram.methods.get\_chat import GetChat
- alias: from aiogram.methods import GetChat

## With specific bot

```
result: ChatFullInfo = await bot(GetChat(...))
```

## getChatAdministrators

Returns: List[Union[ChatMemberOwner, ChatMemberAdministrator, ChatMemberMember, ChatMemberRestricted, ChatMemberLeft, ChatMemberBanned]]

Use this method to get a list of administrators in a chat, which aren't bots. Returns an Array of aiogram.types.chat\_member.ChatMember objects.

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

```
chat_id: int | str
```

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

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### **Usage**

#### As bot method

```
result: List[Union[ChatMemberOwner, ChatMemberAdministrator, ChatMemberMember, ChatMemberRestricted, ChatMemberLeft, ChatMemberBanned]] = await bot.get_chat_administrators(...)
```

## Method as object

#### Imports:

- from aiogram.methods.get\_chat\_administrators import GetChatAdministrators
- alias: from aiogram.methods import GetChatAdministrators

#### With specific bot

```
result: List[Union[ChatMemberOwner, ChatMemberAdministrator, ChatMemberMember, GhatMemberRestricted, ChatMemberLeft, ChatMemberBanned]] = await GetChatAdministrators(...))
```

## As shortcut from received object

aiogram.types.chat.Chat.get\_administrators()

## getChatMember

Returns: Union[ChatMemberOwner, ChatMemberAdministrator, ChatMemberMember, ChatMemberRestricted, ChatMemberLeft, ChatMemberBanned]

Use this method to get information about a member of a chat. The method is only guaranteed to work for other users if the bot is an administrator in the chat. Returns a aiogram.types.chat\_member.ChatMember object on success.

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

```
chat_id: int | str
```

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

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

# user\_id: int

Unique identifier of the target user

#### As bot method

## Method as object

#### Imports:

- from aiogram.methods.get\_chat\_member import GetChatMember
- alias: from aiogram.methods import GetChatMember

# With specific bot

## As shortcut from received object

• aiogram.types.chat.Chat.get\_member()

#### getChatMemberCount

```
Returns: int
```

Use this method to get the number of members in a chat. Returns *Int* on success.

Source: https://core.telegram.org/bots/api#getchatmembercount

```
chat_id: int | str
```

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

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### As bot method

```
result: int = await bot.get_chat_member_count(...)
```

## Method as object

Imports:

- from aiogram.methods.get\_chat\_member\_count import GetChatMemberCount
- alias: from aiogram.methods import GetChatMemberCount

## With specific bot

```
result: int = await bot(GetChatMemberCount(...))
```

# As shortcut from received object

• aiogram.types.chat.Chat.get\_member\_count()

# getChatMenuButton

Returns: Union[MenuButtonDefault, MenuButtonWebApp, MenuButtonCommands]

Use this method to get the current value of the bot's menu button in a private chat, or the default menu button. Returns <code>aiogram.types.menu\_button.MenuButton</code> on success.

Source: https://core.telegram.org/bots/api#getchatmenubutton

```
chat_id: int | None
```

Unique identifier for the target private chat. If not specified, default bot's menu button will be returned

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### As bot method

```
result: Union[MenuButtonDefault, MenuButtonWebApp, MenuButtonCommands] = await bot.get_
—chat_menu_button(...)
```

# Method as object

#### Imports:

- from aiogram.methods.get\_chat\_menu\_button import GetChatMenuButton
- alias: from aiogram.methods import GetChatMenuButton

## With specific bot

```
result: Union[MenuButtonDefault, MenuButtonWebApp, MenuButtonCommands] = await_

substitution = await_

substituti
```

#### getFile

Returns: File

```
class aiogram.methods.get_file.GetFile(*, file_id: str, **extra_data: Any)
```

Use this method to get basic information about a file and prepare it for downloading. For the moment, bots can download files of up to 20MB in size. On success, a <code>aiogram.types.file.File</code> object is returned. The file can then be downloaded via the link <code>https://api.telegram.org/file/bot<token>/<file\_path>, where <file\_path> is taken from the response. 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 <code>aiogram.methods.get\_file.GetFile</code> again. 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.</code>

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

```
file_id: str
```

File identifier to get information about

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### As bot method

```
result: File = await bot.get_file(...)
```

## Method as object

Imports:

- from aiogram.methods.get\_file import GetFile
- alias: from aiogram.methods import GetFile

## With specific bot

```
result: File = await bot(GetFile(...))
```

# getForumTopicIconStickers

Returns: List[Sticker]

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

Source: https://core.telegram.org/bots/api#getforumtopiciconstickers

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### **Usage**

#### As bot method

```
result: List[Sticker] = await bot.get_forum_topic_icon_stickers(...)
```

# Method as object

## Imports:

- from aiogram.methods.get\_forum\_topic\_icon\_stickers import GetForumTopicIconStickers
- alias: from aiogram.methods import GetForumTopicIconStickers

# With specific bot

```
result: List[Sticker] = await bot(GetForumTopicIconStickers(...))
```

## getMe

Returns: User

```
class aiogram.methods.get_me.GetMe(**extra_data: Any)
```

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

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

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

# **Usage**

#### As bot method

```
result: User = await bot.get_me(...)
```

#### Method as object

#### Imports:

- from aiogram.methods.get\_me import GetMe
- alias: from aiogram.methods import GetMe

## With specific bot

```
result: User = await bot(GetMe(...))
```

# getMyCommands

```
Returns: List[BotCommand]
```

 $\textbf{class} \ aiogram. \texttt{methods.get\_my\_commands.GetMyCommands}(*, \textit{scope}: BotCommandScopeDefault | \texttt{methods.get\_my\_commands}(*, \textit{scope}: BotCommandScopeDefault | \texttt{methods.get\_my\_commandScopeDefault}(*, \textit{scope}: BotCommandScopeDefault | \texttt{methods.get\_my\_commandScopeDefault}(*, \textit{scope}: BotCommandScopeDefault}(*, \textitscope: BotCommandScopeDefault}$ 

BotCommandScopeAllPrivateChats | BotCommandScopeAllGroupChats |

BotCommandScopeAllChatAdministrators |

BotCommandScopeChat |

BotCommandScopeChatAdministrators |

BotCommandScopeChatMember | None = None,

 $language\_code: str \mid None = None,$ 

\*\*extra\_data: Any)

Use this method to get the current list of the bot's commands for the given scope and user language. Returns an Array of aiogram.types.bot\_command.BotCommand objects. If commands aren't set, an empty list is returned.

Source: https://core.telegram.org/bots/api#getmycommands

```
scope: BotCommandScopeDefault | BotCommandScopeAllPrivateChats |
BotCommandScopeAllGroupChats | BotCommandScopeAllChatAdministrators |
BotCommandScopeChat | BotCommandScopeChatAdministrators | BotCommandScopeChatMember | None
```

A JSON-serialized object, describing scope of users. Defaults to aiogram.types.bot\_command\_scope\_default.BotCommandScopeDefault.

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
language_code: str | None
```

A two-letter ISO 639-1 language code or an empty string

## **Usage**

## As bot method

```
result: List[BotCommand] = await bot.get_my_commands(...)
```

# Method as object

#### Imports:

- from aiogram.methods.get\_my\_commands import GetMyCommands
- alias: from aiogram.methods import GetMyCommands

#### With specific bot

```
result: List[BotCommand] = await bot(GetMyCommands(...))
```

## getMyDefaultAdministratorRights

Returns: ChatAdministratorRights

class aiogram.methods.get\_my\_default\_administrator\_rights.GetMyDefaultAdministratorRights(\*,

```
for_channels:
bool
|
None
=
None,
**ex-
tra_data:
Any)
```

Use this method to get the current default administrator rights of the bot. Returns aiogram.types.chat\_administrator\_rights.ChatAdministratorRights on success.

Source: https://core.telegram.org/bots/api#getmydefaultadministratorrights

```
for_channels: bool | None
```

Pass True to get default administrator rights of the bot in channels. Otherwise, default administrator rights of the bot for groups and supergroups will be returned.

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### **Usage**

#### As bot method

```
result: ChatAdministratorRights = await bot.get_my_default_administrator_rights(...)
```

## Method as object

#### Imports:

- from aiogram.methods.get\_my\_default\_administrator\_rights import GetMyDefaultAdministratorRights
- alias: from aiogram.methods import GetMyDefaultAdministratorRights

# With specific bot

```
result: ChatAdministratorRights = await bot(GetMyDefaultAdministratorRights(...))
```

## getMyDescription

Returns: BotDescription

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

Source: https://core.telegram.org/bots/api#getmydescription

```
language_code: str | None
```

A two-letter ISO 639-1 language code or an empty string

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

## **Usage**

#### As bot method

```
result: BotDescription = await bot.get_my_description(...)
```

# Method as object

## Imports:

- from aiogram.methods.get\_my\_description import GetMyDescription
- alias: from aiogram.methods import GetMyDescription

# With specific bot

```
result: BotDescription = await bot(GetMyDescription(...))
```

# getMyName

Returns: BotName

class aiogram.methods.get\_my\_name.GetMyName(\*, language\_code: str | None = None, \*\*extra\_data: Any)
 Use this method to get the current bot name for the given user language. Returns aiogram.types.bot\_name.
 BotName on success.

Source: https://core.telegram.org/bots/api#getmyname

language\_code: str | None

A two-letter ISO 639-1 language code or an empty string

model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

 $model_post_init(context: Any, /) \rightarrow None$ 

We need to both initialize private attributes and call the user-defined model\_post\_init method.

### **Usage**

#### As bot method

```
result: BotName = await bot.get_my_name(...)
```

### Method as object

Imports:

- from aiogram.methods.get\_my\_name import GetMyName
- alias: from aiogram.methods import GetMyName

### With specific bot

```
result: BotName = await bot(GetMyName(...))
```

### getMyShortDescription

Returns: BotShortDescription

Use this method to get the current bot short description for the given user language. Returns aiogram.types. bot\_short\_description.BotShortDescription on success.

Source: https://core.telegram.org/bots/api#getmyshortdescription

language\_code: str | None

A two-letter ISO 639-1 language code or an empty string

model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

 $model_post_init(context: Any, /) \rightarrow None$ 

We need to both initialize private attributes and call the user-defined model\_post\_init method.

### **Usage**

#### As bot method

```
result: BotShortDescription = await bot.get_my_short_description(...)
```

### Method as object

Imports:

- from aiogram.methods.get\_my\_short\_description import GetMyShortDescription
- alias: from aiogram.methods import GetMyShortDescription

#### With specific bot

```
result: BotShortDescription = await bot(GetMyShortDescription(...))
```

# getUserChatBoosts

Returns: UserChatBoosts

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

Source: https://core.telegram.org/bots/api#getuserchatboosts

```
chat_id: int | str
```

Unique identifier for the chat or username of the channel (in the format @channelusername)

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### user\_id: int

Unique identifier of the target user

### **Usage**

#### As bot method

```
result: UserChatBoosts = await bot.get_user_chat_boosts(...)
```

### Method as object

Imports:

- from aiogram.methods.get\_user\_chat\_boosts import GetUserChatBoosts
- alias: from aiogram.methods import GetUserChatBoosts

#### With specific bot

```
result: UserChatBoosts = await bot(GetUserChatBoosts(...))
```

### getUserProfilePhotos

Returns: UserProfilePhotos

Use this method to get a list of profile pictures for a user. Returns a aiogram.types.user\_profile\_photos. UserProfilePhotos object.

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

user id: int

Unique identifier of the target user

offset: int | None

Sequential number of the first photo to be returned. By default, all photos are returned.

model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
limit: int | None
```

Limits the number of photos to be retrieved. Values between 1-100 are accepted. Defaults to 100.

#### **Usage**

#### As bot method

```
result: UserProfilePhotos = await bot.get_user_profile_photos(...)
```

### Method as object

Imports:

- from aiogram.methods.get\_user\_profile\_photos import GetUserProfilePhotos
- alias: from aiogram.methods import GetUserProfilePhotos

### With specific bot

```
result: UserProfilePhotos = await bot(GetUserProfilePhotos(...))
```

#### As shortcut from received object

• aiogram.types.user.User.get\_profile\_photos()

### hideGeneralForumTopic

Returns: bool

Use this method to hide the 'General' topic in a forum supergroup chat. The bot must be an administrator in the chat for this to work and must have the *can\_manage\_topics* administrator rights. The topic will be automatically closed if it was open. Returns True on success.

Source: https://core.telegram.org/bots/api#hidegeneralforumtopic

```
chat_id: int | str
```

Unique identifier for the target chat or username of the target supergroup (in the format @supergroupusername)

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### As bot method

```
result: bool = await bot.hide_general_forum_topic(...)
```

# Method as object

Imports:

- from aiogram.methods.hide\_general\_forum\_topic import HideGeneralForumTopic
- alias: from aiogram.methods import HideGeneralForumTopic

### With specific bot

```
result: bool = await bot(HideGeneralForumTopic(...))
```

# As reply into Webhook in handler

```
return HideGeneralForumTopic(...)
```

#### leaveChat

Returns: bool

class aiogram.methods.leave\_chat.LeaveChat(\*, chat\_id: int | str, \*\*extra\_data: Any)

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

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

```
chat_id: int | str
```

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

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### As bot method

```
result: bool = await bot.leave_chat(...)
```

### Method as object

Imports:

- from aiogram.methods.leave\_chat import LeaveChat
- alias: from aiogram.methods import LeaveChat

### With specific bot

```
result: bool = await bot(LeaveChat(...))
```

# As reply into Webhook in handler

```
return LeaveChat(...)
```

### As shortcut from received object

• aiogram.types.chat.Chat.leave()

### logOut

Returns: bool

```
class aiogram.methods.log_out.LogOut(**extra_data: Any)
```

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

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

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### As bot method

```
result: bool = await bot.log_out(...)
```

### Method as object

Imports:

- from aiogram.methods.log\_out import LogOut
- alias: from aiogram.methods import LogOut

### With specific bot

```
result: bool = await bot(LogOut(...))
```

# As reply into Webhook in handler

```
return LogOut(...)
```

### pinChatMessage

Returns: bool

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' administrator right in a supergroup or 'can\_edit\_messages' administrator right in a channel. Returns True on success.

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

```
chat_id: int | str
```

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

```
message_id: int
```

Identifier of a message to pin

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### business\_connection\_id: str | None

Unique identifier of the business connection on behalf of which the message will be pinned

### disable\_notification: bool | None

Pass True if it is not necessary to send a notification to all chat members about the new pinned message. Notifications are always disabled in channels and private chats.

# **Usage**

#### As bot method

```
result: bool = await bot.pin_chat_message(...)
```

# Method as object

#### Imports:

- from aiogram.methods.pin\_chat\_message import PinChatMessage
- alias: from aiogram.methods import PinChatMessage

# With specific bot

```
result: bool = await bot(PinChatMessage(...))
```

### As reply into Webhook in handler

```
return PinChatMessage(...)
```

# As shortcut from received object

- aiogram.types.chat.Chat.pin\_message()
- aiogram.types.message.Message.pin()

# promoteChatMember

Returns: bool

class aiogram.methods.promote\_chat\_member.PromoteChatMember(\*, chat\_id: int | str, user\_id: int,

 $is_anonymous: bool \mid None = None,$ can manage chat: bool | None = None, can\_delete\_messages: bool | None = None,can manage video chats: bool None = None, can restrict members: $bool \mid None = None,$ can promote members: bool | None = None, can\_change\_info: bool | None = None, can\_invite\_users: bool | None = None, can\_post\_stories: bool | None = None, can\_edit\_stories: bool | None = None, can\_delete\_stories: bool | *None* = *None*, *can\_post\_messages*:  $bool \mid None = None,$ can\_edit\_messages: bool | None = None, can pin messages: bool | None = None, can\_manage\_topics: bool | None = None, \*\*extra data: Any)

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 administrator rights. Pass False for all boolean parameters to demote a user. Returns True on success.

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

#### chat\_id: int | str

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

#### user\_id: int

Unique identifier of the target user

### is\_anonymous: bool | None

Pass True if the administrator's presence in the chat is hidden

#### can\_manage\_chat: bool | None

Pass True if the administrator can access the chat event log, get boost list, see hidden supergroup and channel members, report spam messages and ignore slow mode. Implied by any other administrator privilege.

#### can\_delete\_messages: bool | None

Pass True if the administrator can delete messages of other users

#### can\_manage\_video\_chats: bool | None

Pass True if the administrator can manage video chats

#### can\_restrict\_members: bool | None

Pass True if the administrator can restrict, ban or unban chat members, or access supergroup statistics

#### can promote members: bool | None

Pass True if the administrator can add new administrators with a subset of their own privileges or demote administrators that they have promoted, directly or indirectly (promoted by administrators that were appointed by him)

### can\_change\_info: bool | None

Pass True if the administrator can change chat title, photo and other settings

### model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

### $model_post_init(context: Any, /) \rightarrow None$

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### can\_invite\_users: bool | None

Pass True if the administrator can invite new users to the chat

### can\_post\_stories: bool | None

Pass True if the administrator can post stories to the chat

#### can\_edit\_stories: bool | None

Pass True if the administrator can edit stories posted by other users, post stories to the chat page, pin chat stories, and access the chat's story archive

### can\_delete\_stories: bool | None

Pass True if the administrator can delete stories posted by other users

#### can\_post\_messages: bool | None

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

#### can\_edit\_messages: bool | None

Pass True if the administrator can edit messages of other users and can pin messages; for channels only

#### can\_pin\_messages: bool | None

Pass True if the administrator can pin messages; for supergroups only

#### can\_manage\_topics: bool | None

Pass True if the user is allowed to create, rename, close, and reopen forum topics; for supergroups only

### **Usage**

### As bot method

```
result: bool = await bot.promote_chat_member(...)
```

# Method as object

# Imports:

- from aiogram.methods.promote\_chat\_member import PromoteChatMember
- alias: from aiogram.methods import PromoteChatMember

# With specific bot

```
result: bool = await bot(PromoteChatMember(...))
```

# As reply into Webhook in handler

```
return PromoteChatMember(...)
```

### As shortcut from received object

• aiogram.types.chat.Chat.promote()

# reopenForumTopic

Returns: bool

Use this method to reopen a closed topic in a forum supergroup chat. The bot must be an administrator in the chat for this to work and must have the *can\_manage\_topics* administrator rights, unless it is the creator of the topic. Returns True on success.

Source: https://core.telegram.org/bots/api#reopenforumtopic

```
chat_id: int | str
```

Unique identifier for the target chat or username of the target supergroup (in the format @supergroupusername)

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
message_thread_id: int
```

Unique identifier for the target message thread of the forum topic

# **Usage**

# As bot method

```
result: bool = await bot.reopen_forum_topic(...)
```

# Method as object

Imports:

- from aiogram.methods.reopen\_forum\_topic import ReopenForumTopic
- alias: from aiogram.methods import ReopenForumTopic

### With specific bot

```
result: bool = await bot(ReopenForumTopic(...))
```

### As reply into Webhook in handler

```
return ReopenForumTopic(...)
```

### reopenGeneralForumTopic

Returns: bool

Use this method to reopen a closed 'General' topic in a forum supergroup chat. The bot must be an administrator in the chat for this to work and must have the *can\_manage\_topics* administrator rights. The topic will be automatically unhidden if it was hidden. Returns True on success.

Source: https://core.telegram.org/bots/api#reopengeneralforumtopic

```
chat_id: int | str
```

Unique identifier for the target chat or username of the target supergroup (in the format @supergroupusername)

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

# **Usage**

### As bot method

```
result: bool = await bot.reopen_general_forum_topic(...)
```

### Method as object

Imports:

- from aiogram.methods.reopen\_general\_forum\_topic import ReopenGeneralForumTopic
- alias: from aiogram.methods import ReopenGeneralForumTopic

#### With specific bot

```
result: bool = await bot(ReopenGeneralForumTopic(...))
```

### As reply into Webhook in handler

```
return ReopenGeneralForumTopic(...)
```

#### restrictChatMember

Returns: bool

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

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

```
chat_id: int | str
```

Unique identifier for the target chat or username of the target supergroup (in the format @supergroupusername)

```
user_id: int
```

Unique identifier of the target user

```
permissions: ChatPermissions
```

A JSON-serialized object for new user permissions

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
use_independent_chat_permissions: bool | None
```

Pass True if chat permissions are set independently. Otherwise, the *can\_send\_other\_messages* and *can\_add\_web\_page\_previews* permissions will imply the *can\_send\_messages*, *can\_send\_audios*, *can\_send\_documents*, *can\_send\_photos*, *can\_send\_videos*, *can\_send\_video\_notes*, and

can\_send\_voice\_notes permissions; the can\_send\_polls permission will imply the can\_send\_messages permission.

```
until_date: datetime.datetime | datetime.timedelta | int | None
```

Date when restrictions will be lifted for the user; Unix time. If user is restricted for more than 366 days or less than 30 seconds from the current time, they are considered to be restricted forever

### **Usage**

#### As bot method

```
result: bool = await bot.restrict_chat_member(...)
```

# Method as object

#### Imports:

- from aiogram.methods.restrict\_chat\_member import RestrictChatMember
- alias: from aiogram.methods import RestrictChatMember

### With specific bot

```
result: bool = await bot(RestrictChatMember(...))
```

#### As reply into Webhook in handler

```
return RestrictChatMember(...)
```

### As shortcut from received object

• aiogram.types.chat.Chat.restrict()

### revokeChatInviteLink

Returns: ChatInviteLink

Use this method to revoke an invite link created by the bot. If the primary link is revoked, a new link is automatically generated. The bot must be an administrator in the chat for this to work and must have the appropriate administrator rights. Returns the revoked invite link as <code>aiogram.types.chat\_invite\_link.ChatInviteLink</code> object.

Source: https://core.telegram.org/bots/api#revokechatinvitelink

# chat\_id: int | str

Unique identifier of the target chat or username of the target channel (in the format @channelusername)

# model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### invite\_link: str

The invite link to revoke

### **Usage**

#### As bot method

```
result: ChatInviteLink = await bot.revoke_chat_invite_link(...)
```

# Method as object

#### Imports:

- from aiogram.methods.revoke\_chat\_invite\_link import RevokeChatInviteLink
- alias: from aiogram.methods import RevokeChatInviteLink

### With specific bot

```
result: ChatInviteLink = await bot(RevokeChatInviteLink(...))
```

#### As reply into Webhook in handler

```
return RevokeChatInviteLink(...)
```

### As shortcut from received object

• aiogram.types.chat.Chat.revoke\_invite\_link()

#### sendAnimation

Returns: Message

```
class aiogram.methods.send_animation.SendAnimation(*, chat_id: int | str, animation:
```

```
~aiogram.types.input_file.InputFile | str,
business_connection_id: str | None = None,
message\_thread\_id: int \mid None = None, duration:
int \mid None = None, width: int \mid None = None,
height: int \mid None = None, thumbnail:
~aiogram.types.input_file.InputFile | None =
None, caption: str \mid None = None, parse mode:
str | ~aiogram.client.default.Default | None =
< Default('parse mode')>, caption entities: ~typ-
ing.List[~aiogram.types.message entity.MessageEntity]
| None = None, show_caption_above_media: bool
| ~aiogram.client.default.Default | None =
< Default('show_caption_above_media')>,
has\_spoiler: bool \mid None = None,
disable notification: bool \mid None = None,
protect_content: bool |
~aiogram.client.default.Default | None =
< Default('protect_content')>, message_effect_id:
str | None = None, reply_parameters:
~aiogram.types.reply_parameters.ReplyParameters
| None = None, reply markup:
~aiogram.types.inline keyboard markup.InlineKeyboardMarkup
~aiogram.types.reply keyboard markup.ReplyKeyboardMarkup
~aiogram.types.reply keyboard remove.ReplyKeyboardRemove
| ~aiogram.types.force reply.ForceReply | None =
None, allow_sending_without_reply: bool | None
= None, reply_to_message_id: int | None = None,
**extra_data: ~typing.Any)
```

Use this method to send animation files (GIF or H.264/MPEG-4 AVC video without sound). On success, the sent <code>aiogram.types.message.Message</code> 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

### chat\_id: int | str

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

#### animation: InputFile | str

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. *More information on Sending Files* »

### business\_connection\_id: str | None

Unique identifier of the business connection on behalf of which the message will be sent

#### message\_thread\_id: int | None

Unique identifier for the target message thread (topic) of the forum; for forum supergroups only

#### duration: int | None

Duration of sent animation in seconds

# width: int | None

Animation width

# height: int | None

Animation height

#### thumbnail: InputFile | None

Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file\_attach\_name>' if the thumbnail was uploaded using multipart/form-data under <file\_attach\_name>. *More information on Sending Files* »

### caption: str | None

Animation caption (may also be used when resending animation by *file\_id*), 0-1024 characters after entities parsing

```
parse_mode: str | Default | None
```

Mode for parsing entities in the animation caption. See formatting options for more details.

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
caption_entities: List[MessageEntity] | None
```

A JSON-serialized list of special entities that appear in the caption, which can be specified instead of parse\_mode

```
show_caption_above_media: bool | Default | None
```

Pass True, if the caption must be shown above the message media

```
has_spoiler: bool | None
```

Pass True if the animation needs to be covered with a spoiler animation

```
disable_notification: bool | None
```

Sends the message silently. Users will receive a notification with no sound.

```
protect_content: bool | Default | None
```

Protects the contents of the sent message from forwarding and saving

```
message_effect_id: str | None
```

Unique identifier of the message effect to be added to the message; for private chats only

```
reply_parameters: ReplyParameters | None
```

Description of the message to reply to

```
reply_markup: InlineKeyboardMarkup | ReplyKeyboardMarkup | ReplyKeyboardRemove |
ForceReply | None
```

Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

### allow\_sending\_without\_reply: bool | None

Pass True if the message should be sent even if the specified replied-to message is not found

Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023

```
reply_to_message_id: int | None
```

If the message is a reply, ID of the original message

Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023

### **Usage**

#### As bot method

```
result: Message = await bot.send_animation(...)
```

# Method as object

### Imports:

- from aiogram.methods.send\_animation import SendAnimation
- alias: from aiogram.methods import SendAnimation

#### With specific bot

```
result: Message = await bot(SendAnimation(...))
```

### As reply into Webhook in handler

```
return SendAnimation(...)
```

### As shortcut from received object

- aiogram.types.message.Message.answer\_animation()
- aiogram.types.message.Message.reply\_animation()
- aiogram.types.chat\_join\_request.ChatJoinRequest.answer\_animation()
- aiogram.types.chat\_join\_request.ChatJoinRequest.answer\_animation\_pm()
- $\bullet \ aiogram.types.chat\_member\_updated.ChatMemberUpdated.answer\_animation()\\$
- aiogram.types.inaccessible\_message.InaccessibleMessage.answer\_animation()
- aiogram.types.inaccessible\_message.InaccessibleMessage.reply\_animation()

#### sendAudio

```
Returns: Message
```

```
class aiogram.methods.send_audio.SendAudio(*, chat_id: int | str, audio:
```

```
~aiogram.types.input_file.InputFile | str,
business\_connection\_id:\ str\mid None=None,
message_thread_id: int | None = None, caption: str | None =
None, parse_mode: str | ~aiogram.client.default.Default |
None = \langle Default('parse\ mode') \rangle, caption entities:
~typing.List[~aiogram.types.message_entity.MessageEntity]
| None = None, duration: int | None = None, performer: str |
None = None, title: str \mid None = None, thumbnail:
\simaiogram.types.input file.InputFile | None = None,
disable notification: bool | None = None, protect content:
bool | ~aiogram.client.default.Default | None =
<Default('protect_content')>, message_effect_id: str | None
= None, reply_parameters:
~aiogram.types.reply_parameters.ReplyParameters | None =
None, reply markup:
~aiogram.types.inline_keyboard_markup.InlineKeyboardMarkup
~aiogram.types.reply_keyboard_markup.ReplyKeyboardMarkup
~aiogram.types.reply_keyboard_remove.ReplyKeyboardRemove
| \sim aiogram.types.force\_reply.ForceReply | None = None,
allow sending without reply: bool \mid None = None,
reply_to_message_id: int | None = None, **extra_data:
~typing.Any)
```

Use this method to send audio files, if you want Telegram clients to display them in the music player. Your audio must be in the .MP3 or .M4A format. On success, the sent <code>aiogram.types.message.Message</code> is returned. Bots can currently send audio files of up to 50 MB in size, this limit may be changed in the future. For sending voice messages, use the <code>aiogram.methods.send\_voice.SendVoice</code> method instead.

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

#### chat\_id: int | str

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

#### audio: InputFile | str

Audio file to send. Pass a file\_id as String to send an audio file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get an audio file from the Internet, or upload a new one using multipart/form-data. *More information on Sending Files* »

### business\_connection\_id: str | None

Unique identifier of the business connection on behalf of which the message will be sent

### message\_thread\_id: int | None

Unique identifier for the target message thread (topic) of the forum; for forum supergroups only

### caption: str | None

Audio caption, 0-1024 characters after entities parsing

### parse\_mode: str | Default | None

Mode for parsing entities in the audio caption. See formatting options for more details.

#### caption\_entities: List[MessageEntity] | None

A JSON-serialized list of special entities that appear in the caption, which can be specified instead of parse\_mode

duration: int | None

Duration of the audio in seconds

performer: str | None

Performer

#### model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

title: str | None

Track name

#### thumbnail: InputFile | None

Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file\_attach\_name>' if the thumbnail was uploaded using multipart/form-data under <file\_attach\_name>. *More information on Sending Files* »

### disable\_notification: bool | None

Sends the message silently. Users will receive a notification with no sound.

```
protect_content: bool | Default | None
```

Protects the contents of the sent message from forwarding and saving

```
message_effect_id: str | None
```

Unique identifier of the message effect to be added to the message; for private chats only

```
reply_parameters: ReplyParameters | None
```

Description of the message to reply to

```
reply_markup: InlineKeyboardMarkup | ReplyKeyboardMarkup | ReplyKeyboardRemove |
ForceReply | None
```

Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

```
allow_sending_without_reply: bool | None
```

Pass True if the message should be sent even if the specified replied-to message is not found

Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023

### reply\_to\_message\_id: int | None

If the message is a reply, ID of the original message

Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023

#### As bot method

```
result: Message = await bot.send_audio(...)
```

### Method as object

Imports:

- from aiogram.methods.send\_audio import SendAudio
- alias: from aiogram.methods import SendAudio

### With specific bot

```
result: Message = await bot(SendAudio(...))
```

# As reply into Webhook in handler

```
return SendAudio(...)
```

### As shortcut from received object

- aiogram.types.message.Message.answer\_audio()
- aiogram.types.message.Message.reply\_audio()
- aiogram.types.chat\_join\_request.ChatJoinRequest.answer\_audio()
- aiogram.types.chat\_join\_request.ChatJoinRequest.answer\_audio\_pm()
- aiogram.types.chat\_member\_updated.ChatMemberUpdated.answer\_audio()
- aiogram.types.inaccessible\_message.InaccessibleMessage.answer\_audio()
- aiogram.types.inaccessible\_message.InaccessibleMessage.reply\_audio()

### sendChatAction

Returns: bool

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

Example: The ImageBot needs some time to process a request and upload the image. Instead of sending a text message along the lines of 'Retrieving image, please wait...', the bot may use <code>aiogram.methods.send\_chat\_action.SendChatAction</code> with <code>action = upload\_photo</code>. The user will see a 'sending photo' status for the bot.

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

```
chat_id: int | str
```

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

action: str

Type of action to broadcast. Choose one, depending on what the user is about to receive: typing for text messages, upload\_photo for photos, record\_video or upload\_video for videos, record\_voice or upload\_voice for voice notes, upload\_document for general files, choose\_sticker for stickers, find\_location for location data, record\_video\_note or upload\_video\_note for video notes.

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
business_connection_id: str | None
```

Unique identifier of the business connection on behalf of which the action will be sent

```
message_thread_id: int | None
```

Unique identifier for the target message thread; for supergroups only

#### **Usage**

#### As bot method

```
result: bool = await bot.send_chat_action(...)
```

### Method as object

Imports:

- from aiogram.methods.send\_chat\_action import SendChatAction
- alias: from aiogram.methods import SendChatAction

# With specific bot

```
result: bool = await bot(SendChatAction(...))
```

# As reply into Webhook in handler

```
return SendChatAction(...)
```

### As shortcut from received object

• aiogram.types.chat.Chat.do()

#### sendContact

```
Returns: Message
```

```
class aiogram.methods.send_contact.SendContact(*, chat_id: int | str, phone_number: str, first_name:
                                                          str, business_connection_id: str | None = None,
                                                          message_thread_id: int | None = None, last_name: str |
                                                          None = None, vcard: str \mid None = None,
                                                          disable\_notification: bool \mid None = None,
                                                          protect_content: bool | ~aiogram.client.default.Default
                                                          | None = < Default('protect_content')>,
                                                          message\_effect\_id: str \mid None = None,
                                                          reply parameters:
                                                          ~aiogram.types.reply_parameters.ReplyParameters |
                                                          None = None, reply markup:
                                                          ~aiogram.types.inline_keyboard_markup.InlineKeyboardMarkup
                                                          ~aiogram.types.reply_keyboard_markup.ReplyKeyboardMarkup
                                                          ~aiogram.types.reply_keyboard_remove.ReplyKeyboardRemove
                                                          | ~aiogram.types.force_reply.ForceReply | None =
                                                          None, allow_sending_without_reply: bool | None =
                                                          None, reply\_to\_message\_id: int | None = None,
                                                          **extra_data: ~typing.Any)
```

Use this method to send phone contacts. On success, the sent aiogram.types.message.Message is returned.

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

```
chat_id: int | str
```

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

phone\_number: str

Contact's phone number

first\_name: str

Contact's first name

#### business\_connection\_id: str | None

Unique identifier of the business connection on behalf of which the message will be sent

### message\_thread\_id: int | None

Unique identifier for the target message thread (topic) of the forum; for forum supergroups only

#### last\_name: str | None

Contact's last name

### vcard: str | None

Additional data about the contact in the form of a vCard, 0-2048 bytes

### model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### disable\_notification: bool | None

Sends the message silently. Users will receive a notification with no sound.

```
protect_content: bool | Default | None
```

Protects the contents of the sent message from forwarding and saving

```
message_effect_id: str | None
```

Unique identifier of the message effect to be added to the message; for private chats only

```
reply_parameters: ReplyParameters | None
```

Description of the message to reply to

```
reply_markup: InlineKeyboardMarkup | ReplyKeyboardMarkup | ReplyKeyboardRemove |
ForceReply | None
```

Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

```
allow_sending_without_reply: bool | None
```

Pass True if the message should be sent even if the specified replied-to message is not found

Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023

```
reply_to_message_id: int | None
```

If the message is a reply, ID of the original message

Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023

### **Usage**

## As bot method

```
result: Message = await bot.send_contact(...)
```

# Method as object

#### Imports:

- from aiogram.methods.send\_contact import SendContact
- alias: from aiogram.methods import SendContact

### With specific bot

```
result: Message = await bot(SendContact(...))
```

### As reply into Webhook in handler

```
return SendContact(...)
```

# As shortcut from received object

```
• aiogram.types.message.Message.answer_contact()
```

- aiogram.types.message.Message.reply\_contact()
- aiogram.types.chat\_join\_request.ChatJoinRequest.answer\_contact()
- aiogram.types.chat\_join\_request.ChatJoinRequest.answer\_contact\_pm()
- aiogram.types.chat\_member\_updated.ChatMemberUpdated.answer\_contact()
- aiogram.types.inaccessible\_message.InaccessibleMessage.answer\_contact()
- aiogram.types.inaccessible\_message.InaccessibleMessage.reply\_contact()

#### sendDice

### Returns: Message

```
class aiogram.methods.send_dice.SendDice(*, chat_id: int | str, business_connection_id: str | None =

None, message_thread_id: int | None = None, emoji: str | None

= None, disable_notification: bool | None = None,
protect_content: bool | ~aiogram.client.default.Default | None

= <Default('protect_content')>, message_effect_id: str | None

= None, reply_parameters:
~aiogram.types.reply_parameters.ReplyParameters | None =
None, reply_markup:
~aiogram.types.inline_keyboard_markup.InlineKeyboardMarkup

|
~aiogram.types.reply_keyboard_markup.ReplyKeyboardMarkup
|
~aiogram.types.reply_keyboard_remove.ReplyKeyboardRemove
| ~aiogram.types.force_reply.ForceReply | None = None,
allow_sending_without_reply: bool | None = None,
reply_to_message_id: int | None = None, **extra_data:
~typing.Any)
```

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

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

```
chat_id: int | str
```

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

```
business_connection_id: str | None
```

Unique identifier of the business connection on behalf of which the message will be sent

```
message_thread_id: int | None
```

Unique identifier for the target message thread (topic) of the forum; for forum supergroups only

```
emoji: str | None
```

Emoji on which the dice throw animation is based. Currently, must be one of ", ", ", ", ", or ". Dice can have values 1-6 for ", " and ", values 1-5 for " and ", and values 1-64 for ". Defaults to "

```
disable_notification: bool | None
```

Sends the message silently. Users will receive a notification with no sound.

```
protect_content: bool | Default | None
```

Protects the contents of the sent message from forwarding

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
message_effect_id: str | None
```

Unique identifier of the message effect to be added to the message; for private chats only

```
reply_parameters: ReplyParameters | None
```

Description of the message to reply to

```
reply_markup: InlineKeyboardMarkup | ReplyKeyboardMarkup | ReplyKeyboardRemove |
ForceReply | None
```

Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

```
allow_sending_without_reply: bool | None
```

Pass True if the message should be sent even if the specified replied-to message is not found

Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023

```
reply_to_message_id: int | None
```

452

If the message is a reply, ID of the original message

Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023

#### As bot method

```
result: Message = await bot.send_dice(...)
```

# Method as object

### Imports:

- from aiogram.methods.send\_dice import SendDice
- alias: from aiogram.methods import SendDice

### With specific bot

```
result: Message = await bot(SendDice(...))
```

# As reply into Webhook in handler

```
return SendDice(...)
```

## As shortcut from received object

- aiogram.types.message.Message.answer\_dice()
- aiogram.types.message.Message.reply\_dice()
- aiogram.types.chat\_join\_request.ChatJoinRequest.answer\_dice()
- aiogram.types.chat\_join\_request.ChatJoinRequest.answer\_dice\_pm()
- aiogram.types.chat\_member\_updated.ChatMemberUpdated.answer\_dice()
- aiogram.types.inaccessible\_message.InaccessibleMessage.answer\_dice()
- aiogram.types.inaccessible\_message.InaccessibleMessage.reply\_dice()

# sendDocument

Returns: Message

**class** aiogram.methods.send\_document.**SendDocument**(\*, chat\_id: int | str, document:

```
~aiogram.types.input_file.InputFile | str,
business connection id: str \mid None = None,
message_thread_id: int | None = None, thumbnail:
\simaiogram.types.input file.InputFile | None = None,
caption: str \mid None = None, parse mode: str \mid
~aiogram.client.default.Default | None =
<Default('parse mode')>, caption entities: ~typ-
ing.List[~aiogram.types.message entity.MessageEntity]
| None = None, disable_content_type_detection:
bool | None = None, disable_notification: bool |
None = None, protect\_content: bool
~aiogram.client.default.Default | None =
<Default('protect_content')>, message_effect_id: str
| None = None, reply_parameters:
~aiogram.types.reply_parameters.ReplyParameters |
None = None, reply_markup:
~aiogram.types.inline keyboard markup.InlineKeyboardMarkup
~aiogram.types.reply keyboard markup.ReplyKeyboardMarkup
~aiogram.types.reply_keyboard_remove.ReplyKeyboardRemove
| ~aiogram.types.force_reply.ForceReply | None =
None, allow sending without reply: bool | None =
None, reply to message id: int \mid None = None,
**extra data: ~typing.Any)
```

Use this method to send general files. On success, the sent aiogram.types.message.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

### chat\_id: int | str

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

#### document: InputFile | str

File to send. Pass a file\_id as String to send a file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a file from the Internet, or upload a new one using multipart/form-data. *More information on Sending Files* »

### business\_connection\_id: str | None

Unique identifier of the business connection on behalf of which the message will be sent

# message\_thread\_id: int | None

Unique identifier for the target message thread (topic) of the forum; for forum supergroups only

#### thumbnail: InputFile | None

Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file\_attach\_name>' if the thumbnail was uploaded using multipart/form-data under <file\_attach\_name>. *More information on Sending Files* »

#### caption: str | None

Document caption (may also be used when resending documents by *file\_id*), 0-1024 characters after entities parsing

#### parse\_mode: str | Default | None

Mode for parsing entities in the document caption. See formatting options for more details.

#### caption\_entities: List[MessageEntity] | None

A JSON-serialized list of special entities that appear in the caption, which can be specified instead of parse\_mode

### model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
disable_content_type_detection: bool | None
```

Disables automatic server-side content type detection for files uploaded using multipart/form-data

```
disable_notification: bool | None
```

Sends the message silently. Users will receive a notification with no sound.

```
protect_content: bool | Default | None
```

Protects the contents of the sent message from forwarding and saving

```
message_effect_id: str | None
```

Unique identifier of the message effect to be added to the message; for private chats only

```
reply_parameters: ReplyParameters | None
```

Description of the message to reply to

```
reply_markup: InlineKeyboardMarkup | ReplyKeyboardMarkup | ReplyKeyboardRemove |
ForceReply | None
```

Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

```
allow_sending_without_reply: bool | None
```

Pass True if the message should be sent even if the specified replied-to message is not found

Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023

```
reply_to_message_id: int | None
```

If the message is a reply, ID of the original message

Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023

### **Usage**

#### As bot method

```
result: Message = await bot.send_document(...)
```

# Method as object

### Imports:

- from aiogram.methods.send\_document import SendDocument
- alias: from aiogram.methods import SendDocument

### With specific bot

```
result: Message = await bot(SendDocument(...))
```

### As reply into Webhook in handler

```
return SendDocument(...)
```

# As shortcut from received object

- aiogram.types.message.Message.answer\_document()
- aiogram.types.message.Message.reply\_document()
- aiogram.types.chat\_join\_request.ChatJoinRequest.answer\_document()
- aiogram.types.chat\_join\_request.ChatJoinRequest.answer\_document\_pm()
- aiogram.types.chat\_member\_updated.ChatMemberUpdated.answer\_document()
- aiogram.types.inaccessible\_message.InaccessibleMessage.answer\_document()
- aiogram.types.inaccessible\_message.InaccessibleMessage.reply\_document()

#### sendLocation

Returns: Message

```
class aiogram.methods.send_location.SendLocation(*, chat id: int | str, latitude: float, longitude: float,
                                                             business connection id: str \mid None = None,
                                                             message thread id: int | None = None,
                                                             horizontal\_accuracy: float \mid None = None,
                                                             live period: int | None = None, heading: int | None
                                                             = None, proximity alert radius: int | None = None,
                                                             disable notification: bool \mid None = None,
                                                             protect content: bool |
                                                             ~aiogram.client.default.Default | None =
                                                             <Default('protect_content')>, message_effect_id: str
                                                             | None = None, reply_parameters:
                                                             ~aiogram.types.reply_parameters.ReplyParameters |
                                                             None = None, reply\_markup:
                                                             ~aiogram.types.inline_keyboard_markup.InlineKeyboardMarkup
                                                             ~aiogram.types.reply_keyboard_markup.ReplyKeyboardMarkup
                                                             ~aiogram.types.reply keyboard remove.ReplyKeyboardRemove
                                                             | ~aiogram.types.force_reply.ForceReply | None =
                                                             None, allow sending without reply: bool | None =
                                                             None, reply\_to\_message\_id: int | None = None,
                                                             **extra_data: ~typing.Any)
```

Use this method to send point on the map. On success, the sent aiogram. types.message.Message is returned.

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

#### chat\_id: int | str

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

#### latitude: float

Latitude of the location

#### longitude: float

Longitude of the location

#### business\_connection\_id: str | None

Unique identifier of the business connection on behalf of which the message will be sent

### message\_thread\_id: int | None

Unique identifier for the target message thread (topic) of the forum; for forum supergroups only

#### horizontal\_accuracy: float | None

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

### live\_period: int | None

Period in seconds during which the location will be updated (see Live Locations, should be between 60 and 86400, or 0x7FFFFFF for live locations that can be edited indefinitely.

#### heading: int | None

For live locations, a direction in which the user is moving, in degrees. Must be between 1 and 360 if specified.

#### model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

#### $model_post_init(context: Any, /) \rightarrow None$

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### proximity\_alert\_radius: int | None

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: bool | None

Sends the message silently. Users will receive a notification with no sound.

```
protect_content: bool | Default | None
```

Protects the contents of the sent message from forwarding and saving

```
message_effect_id: str | None
```

Unique identifier of the message effect to be added to the message; for private chats only

```
reply_parameters: ReplyParameters | None
```

Description of the message to reply to

```
reply_markup: InlineKeyboardMarkup | ReplyKeyboardMarkup | ReplyKeyboardRemove |
ForceReply | None
```

Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

```
allow_sending_without_reply: bool | None
```

Pass True if the message should be sent even if the specified replied-to message is not found

Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023

```
reply_to_message_id: int | None
```

If the message is a reply, ID of the original message

Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023

### **Usage**

#### As bot method

```
result: Message = await bot.send_location(...)
```

#### Method as object

#### Imports:

- from aiogram.methods.send\_location import SendLocation
- alias: from aiogram.methods import SendLocation

### With specific bot

```
result: Message = await bot(SendLocation(...))
```

# As reply into Webhook in handler

```
return SendLocation(...)
```

### As shortcut from received object

- aiogram.types.message.Message.answer\_location()
- aiogram.types.message.Message.reply\_location()
- aiogram.types.chat\_join\_request.ChatJoinRequest.answer\_location()
- aiogram.types.chat\_join\_request.ChatJoinRequest.answer\_location\_pm()
- aiogram.types.chat\_member\_updated.ChatMemberUpdated.answer\_location()
- aiogram.types.inaccessible\_message.InaccessibleMessage.answer\_location()
- aiogram.types.inaccessible\_message.InaccessibleMessage.reply\_location()

### sendMediaGroup

```
Returns: List[Message]
```

```
class aiogram.methods.send_media_group.SendMediaGroup(*, chat_id: int | str, media: ~typ-
                                                                    ing.List[~aiogram.types.input media audio.InputMediaAudio
                                                                    ~aiogram.types.input_media_document.InputMediaDocument
                                                                    ~aiogram.types.input_media_photo.InputMediaPhoto
                                                                   ~aiogram.types.input_media_video.InputMediaVideo],
                                                                   business\_connection\_id: str \mid None = None,
                                                                   message\_thread\_id: int \mid None = None,
                                                                   disable\_notification: bool \mid None = None,
                                                                   protect content: bool |
                                                                    ~aiogram.client.default.Default | None =
                                                                    < Default('protect_content')>,
                                                                   message_effect_id: str | None = None,
                                                                   reply_parameters:
                                                                    ~aiogram.types.reply_parameters.ReplyParameters
                                                                    | None = None, allow_sending_without_reply:
                                                                   bool \mid None = None, reply to message id: int
                                                                   | None = None, **extra_data: \sim typing.Any)
```

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

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

#### chat\_id: int | str

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

media: List[InputMediaAudio | InputMediaDocument | InputMediaPhoto |
InputMediaVideo]

A JSON-serialized array describing messages to be sent, must include 2-10 items

#### business\_connection\_id: str | None

Unique identifier of the business connection on behalf of which the message will be sent

### message\_thread\_id: int | None

Unique identifier for the target message thread (topic) of the forum; for forum supergroups only

#### disable\_notification: bool | None

Sends messages silently. Users will receive a notification with no sound.

# model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

# protect\_content: bool | Default | None

Protects the contents of the sent messages from forwarding and saving

#### message\_effect\_id: str | None

Unique identifier of the message effect to be added to the message; for private chats only

### reply\_parameters: ReplyParameters | None

Description of the message to reply to

#### allow\_sending\_without\_reply: bool | None

Pass True if the message should be sent even if the specified replied-to message is not found

Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023

#### reply\_to\_message\_id: int | None

If the messages are a reply, ID of the original message

Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023

#### Usage

#### As bot method

```
result: List[Message] = await bot.send_media_group(...)
```

# Method as object

### Imports:

- from aiogram.methods.send\_media\_group import SendMediaGroup
- alias: from aiogram.methods import SendMediaGroup

### With specific bot

```
result: List[Message] = await bot(SendMediaGroup(...))
```

### As reply into Webhook in handler

```
return SendMediaGroup(...)
```

# As shortcut from received object

- aiogram.types.message.Message.answer\_media\_group()
- aiogram.types.message.Message.reply\_media\_group()
- aiogram.types.chat\_join\_request.ChatJoinRequest.answer\_media\_group()
- aiogram.types.chat\_join\_request.ChatJoinRequest.answer\_media\_group\_pm()
- aiogram.types.chat\_member\_updated.ChatMemberUpdated.answer\_media\_group()
- aiogram.types.inaccessible\_message.InaccessibleMessage.answer\_media\_group()
- aiogram.types.inaccessible\_message.InaccessibleMessage.reply\_media\_group()

### sendMessage

Returns: Message

```
class aiogram.methods.send_message.SendMessage(*, chat id: int | str, text: str, business connection id:
                                                         str | None = None, message_thread_id: int | None =
                                                         None, parse mode: str |
                                                         ~aiogram.client.default.Default | None =
                                                         <Default('parse mode')>, entities: ~typ-
                                                         ing.List[~aiogram.types.message entity.MessageEntity]
                                                         | None = None, link preview options:
                                                         ~aiogram.types.link_preview_options.LinkPreviewOptions
                                                         | ~aiogram.client.default.Default | None =
                                                         <Default('link_preview')>, disable_notification: bool |
                                                         None = None, protect\_content: bool
                                                         ~aiogram.client.default.Default | None =
                                                         <Default('protect_content')>, message_effect_id: str |
                                                         None = None, reply_parameters:
                                                         ~aiogram.types.reply_parameters.ReplyParameters |
                                                         None = None, reply_markup:
                                                         ~aiogram.types.inline_keyboard_markup.InlineKeyboardMarkup
                                                         ~aiogram.types.reply keyboard markup.ReplyKeyboardMarkup
                                                         ~aiogram.types.reply_keyboard_remove.ReplyKeyboardRemove
                                                         | ~aiogram.types.force reply.ForceReply | None =
                                                         None, allow_sending_without_reply: bool | None =
                                                         None, disable web page preview: bool |
                                                         ~aiogram.client.default.Default | None =
                                                         <Default('link preview is disabled')>,
                                                         reply_to_message_id: int | None = None,
                                                         **extra_data: ~typing.Any)
     Use this method to send text messages. On success, the sent aiogram.types.message.Message is returned.
     Source: https://core.telegram.org/bots/api#sendmessage
     chat_id: int | str
           Unique identifier for the target chat or username of the target channel (in the format @channelusername)
     text:
           Text of the message to be sent, 1-4096 characters after entities parsing
     business_connection_id: str | None
           Unique identifier of the business connection on behalf of which the message will be sent
     message_thread_id: int | None
           Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
     parse_mode: str | Default | None
           Mode for parsing entities in the message text. See formatting options for more details.
     entities: List[MessageEntity] | None
           A JSON-serialized list of special entities that appear in message text, which can be specified instead of
           parse_mode
     link_preview_options: LinkPreviewOptions | Default | None
           Link preview generation options for the message
     disable_notification: bool | None
           Sends the message silently. Users will receive a notification with no sound.
```

## model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
protect_content: bool | Default | None
```

Protects the contents of the sent message from forwarding and saving

```
message_effect_id: str | None
```

Unique identifier of the message effect to be added to the message; for private chats only

```
reply_parameters: ReplyParameters | None
```

Description of the message to reply to

```
reply_markup: InlineKeyboardMarkup | ReplyKeyboardMarkup | ReplyKeyboardMemove |
ForceReply | None
```

Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

```
allow_sending_without_reply: bool | None
```

Pass True if the message should be sent even if the specified replied-to message is not found

Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023

```
disable_web_page_preview: bool | Default | None
```

Disables link previews for links in this message

Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023

```
reply_to_message_id: int | None
```

If the message is a reply, ID of the original message

Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023

#### **Usage**

## As bot method

```
result: Message = await bot.send_message(...)
```

# Method as object

# Imports:

- from aiogram.methods.send\_message import SendMessage
- alias: from aiogram.methods import SendMessage

## With specific bot

```
result: Message = await bot(SendMessage(...))
```

## As reply into Webhook in handler

```
return SendMessage(...)
```

## As shortcut from received object

- aiogram.types.message.Message.answer()
- aiogram.types.message.Message.reply()
- aiogram.types.chat\_join\_request.ChatJoinRequest.answer()
- aiogram.types.chat\_join\_request.ChatJoinRequest.answer\_pm()
- aiogram.types.chat\_member\_updated.ChatMemberUpdated.answer()
- aiogram.types.inaccessible\_message.InaccessibleMessage.answer()
- aiogram.types.inaccessible\_message.InaccessibleMessage.reply()

#### sendPaidMedia

Returns: Message

class aiogram.methods.send\_paid\_media.SendPaidMedia(\*, chat id: int | str, star count: int, media:

List[InputPaidMediaPhoto |
InputPaidMediaVideo], business\_connection\_id:
str | None = None, payload: str | None = None,
caption: str | None = None, parse\_mode: str |
None = None, caption\_entities:
List[MessageEntity] | None = None,
show\_caption\_above\_media: bool | None =
None, disable\_notification: bool | None = None,
protect\_content: bool | None = None,
reply\_parameters: ReplyParameters | None =
None, reply\_markup: InlineKeyboardMarkup |
ReplyKeyboardMarkup | ReplyKeyboardRemove

| ForceReply | None = None, \*\*extra data: Any)

Use this method to send paid media. On success, the sent aiogram.types.message.Message is returned.

Source: https://core.telegram.org/bots/api#sendpaidmedia

# chat\_id: int | str

Unique identifier for the target chat or username of the target channel (in the format @channelusername). If the chat is a channel, all Telegram Star proceeds from this media will be credited to the chat's balance. Otherwise, they will be credited to the bot's balance.

#### star\_count: int

The number of Telegram Stars that must be paid to buy access to the media; 1-2500

## media: List[InputPaidMediaPhoto | InputPaidMediaVideo]

A JSON-serialized array describing the media to be sent; up to 10 items

#### business\_connection\_id: str | None

Unique identifier of the business connection on behalf of which the message will be sent

#### payload: str | None

Bot-defined paid media payload, 0-128 bytes. This will not be displayed to the user, use it for your internal processes.

#### caption: str | None

Media caption, 0-1024 characters after entities parsing

#### parse\_mode: str | None

Mode for parsing entities in the media caption. See formatting options for more details.

#### model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
caption_entities: List[MessageEntity] | None
```

A JSON-serialized list of special entities that appear in the caption, which can be specified instead of parse\_mode

#### show\_caption\_above\_media: bool | None

Pass True, if the caption must be shown above the message media

#### disable\_notification: bool | None

Sends the message silently. Users will receive a notification with no sound.

## protect\_content: bool | None

Protects the contents of the sent message from forwarding and saving

#### reply\_parameters: ReplyParameters | None

Description of the message to reply to

```
reply_markup: InlineKeyboardMarkup | ReplyKeyboardMarkup | ReplyKeyboardRemove | ForceReply | None
```

Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

## **Usage**

## As bot method

```
result: Message = await bot.send_paid_media(...)
```

# Method as object

# Imports:

- from aiogram.methods.send\_paid\_media import SendPaidMedia
- alias: from aiogram.methods import SendPaidMedia

# With specific bot

```
result: Message = await bot(SendPaidMedia(...))
```

# As reply into Webhook in handler

```
return SendPaidMedia(...)
```

# As shortcut from received object

- aiogram.types.message.Message.answer\_paid\_media()
- aiogram.types.message.Message.reply\_paid\_media()
- aiogram.types.inaccessible\_message.InaccessibleMessage.answer\_paid\_media()
- aiogram.types.inaccessible\_message.InaccessibleMessage.reply\_paid\_media()

# sendPhoto

Returns: Message

class aiogram.methods.send\_photo.SendPhoto(\*, chat\_id: int | str, photo:

```
~aiogram.types.input_file.InputFile | str,
business connection id: str \mid None = None,
message_thread_id: int | None = None, caption: str | None =
None, parse mode: str | ~aiogram.client.default.Default |
None = <Default('parse mode')>, caption entities:
~typing.List[~aiogram.types.message entity.MessageEntity]
| None = None, show caption above media: bool |
~aiogram.client.default.Default | None =
<Default('show_caption_above_media')>, has_spoiler: bool
| None = None, disable\_notification: bool | None = None,
protect_content: bool | ~aiogram.client.default.Default |
None = <Default('protect_content')>, message_effect_id: str
|None = None, reply_parameters:
~aiogram.types.reply_parameters.ReplyParameters | None =
None, reply_markup:
~aiogram.types.inline_keyboard_markup.InlineKeyboardMarkup
~aiogram.types.reply_keyboard_markup.ReplyKeyboardMarkup
~aiogram.types.reply_keyboard_remove.ReplyKeyboardRemove
| \sim aiogram.types.force reply.ForceReply | None = None,
allow\_sending\_without\_reply: bool | None = None,
reply to message id: int | None = None, **extra data:
~typing.Any)
```

Use this method to send photos. On success, the sent aiogram. types.message.Message is returned.

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

## chat\_id: int | str

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

#### photo: InputFile | str

Photo to send. Pass a file\_id as String to send a photo that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a photo from the Internet, or upload a new photo using multipart/form-data. The photo must be at most 10 MB in size. The photo's width and height must not exceed 10000 in total. Width and height ratio must be at most 20. *More information on Sending Files* »

#### business\_connection\_id: str | None

Unique identifier of the business connection on behalf of which the message will be sent

#### message\_thread\_id: int | None

Unique identifier for the target message thread (topic) of the forum; for forum supergroups only

## caption: str | None

Photo caption (may also be used when resending photos by file\_id), 0-1024 characters after entities parsing

#### parse\_mode: str | Default | None

Mode for parsing entities in the photo caption. See formatting options for more details.

#### caption\_entities: List[MessageEntity] | None

A JSON-serialized list of special entities that appear in the caption, which can be specified instead of <code>parse\_mode</code>

#### show\_caption\_above\_media: bool | Default | None

Pass True, if the caption must be shown above the message media

## model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### has\_spoiler: bool | None

Pass True if the photo needs to be covered with a spoiler animation

#### disable\_notification: bool | None

Sends the message silently. Users will receive a notification with no sound.

```
protect_content: bool | Default | None
```

Protects the contents of the sent message from forwarding and saving

```
message_effect_id: str | None
```

Unique identifier of the message effect to be added to the message; for private chats only

```
reply_parameters: ReplyParameters | None
```

Description of the message to reply to

```
reply_markup: InlineKeyboardMarkup | ReplyKeyboardMarkup | ReplyKeyboardRemove |
ForceReply | None
```

Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

```
allow_sending_without_reply: bool | None
```

Pass True if the message should be sent even if the specified replied-to message is not found

Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023

```
reply_to_message_id: int | None
```

If the message is a reply, ID of the original message

Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023

#### **Usage**

#### As bot method

```
result: Message = await bot.send_photo(...)
```

# Method as object

# Imports:

- from aiogram.methods.send\_photo import SendPhoto
- alias: from aiogram.methods import SendPhoto

# With specific bot

```
result: Message = await bot(SendPhoto(...))
```

# As reply into Webhook in handler

```
return SendPhoto(...)
```

# As shortcut from received object

- aiogram.types.message.Message.answer\_photo()
- aiogram.types.message.Message.reply\_photo()
- aiogram.types.chat\_join\_request.ChatJoinRequest.answer\_photo()
- aiogram.types.chat\_join\_request.ChatJoinRequest.answer\_photo\_pm()
- aiogram.types.chat\_member\_updated.ChatMemberUpdated.answer\_photo()
- aiogram.types.inaccessible\_message.InaccessibleMessage.answer\_photo()
- $\bullet \ aiogram.types.inaccessible\_message.InaccessibleMessage.reply\_photo()\\$

#### sendPoll

Returns: Message

```
ing.List[~aiogram.types.input_poll_option.InputPollOption |
                                             str], business connection id: str | None = None,
                                             message_thread_id: int | None = None, question_parse_mode:
                                             str | ~aiogram.client.default.Default | None =
                                             < Default('parse mode')>, question entities:
                                             ~typing.List[~aiogram.types.message entity.MessageEntity] |
                                             None = None, is anonymous: bool | None = None, type: str |
                                             None = None, allows\_multiple\_answers: bool \mid None = None,
                                             correct_option_id: int | None = None, explanation: str | None
                                             = None, explanation_parse_mode: str |
                                             ~aiogram.client.default.Default | None =
                                             <Default('parse_mode')>, explanation_entities:
                                             ~typing.List[~aiogram.types.message_entity.MessageEntity] |
                                             None = None, open_period: int | None = None, close_date:
                                             \simdatetime.datetime | \simdatetime.timedelta | int | None = None,
                                             is_closed: bool | None = None, disable_notification: bool |
                                             None = None, protect content: bool |
                                             ~aiogram.client.default.Default | None =
                                             <Default('protect_content')>, message_effect_id: str | None =
                                             None, reply_parameters:
                                             ~aiogram.types.reply_parameters.ReplyParameters | None =
                                             None, reply_markup:
                                             ~aiogram.types.inline keyboard markup.InlineKeyboardMarkup
                                             ~aiogram.types.reply_keyboard_markup.ReplyKeyboardMarkup
                                             ~aiogram.types.reply_keyboard_remove.ReplyKeyboardRemove
                                             | ~aiogram.types.force_reply.ForceReply | None = None,
                                             allow\_sending\_without\_reply: bool | None = None,
                                             reply_to_message_id: int | None = None, **extra_data:
                                             ~typing.Any)
Use this method to send a native poll. On success, the sent aiogram. types.message.Message is returned.
Source: https://core.telegram.org/bots/api#sendpoll
chat_id: int | str
     Unique identifier for the target chat or username of the target channel (in the format @channelusername)
question: str
     Poll question, 1-300 characters
options: List[InputPollOption | str]
     A JSON-serialized list of 2-10 answer options
business_connection_id: str | None
     Unique identifier of the business connection on behalf of which the message will be sent
message_thread_id: int | None
     Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
question_parse_mode: str | Default | None
     Mode for parsing entities in the question. See formatting options for more details. Currently, only custom
     emoji entities are allowed
```

**class** aiogram.methods.send\_poll.**SendPoll**(\*, chat id: int | str, question: str, options: ~typ-

#### question\_entities: List[MessageEntity] | None

A JSON-serialized list of special entities that appear in the poll question. It can be specified instead of question\_parse\_mode

#### is\_anonymous: bool | None

True, if the poll needs to be anonymous, defaults to True

## type: str | None

Poll type, 'quiz' or 'regular', defaults to 'regular'

#### allows\_multiple\_answers: bool | None

True, if the poll allows multiple answers, ignored for polls in quiz mode, defaults to False

#### correct\_option\_id: int | None

0-based identifier of the correct answer option, required for polls in quiz mode

#### explanation: str | None

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

## model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

# $model_post_init(context: Any, /) \rightarrow None$

We need to both initialize private attributes and call the user-defined model\_post\_init method.

## explanation\_parse\_mode: str | Default | None

Mode for parsing entities in the explanation. See formatting options for more details.

#### explanation\_entities: List[MessageEntity] | None

A JSON-serialized list of special entities that appear in the poll explanation. It can be specified instead of *explanation\_parse\_mode* 

#### open\_period: int | None

Amount of time in seconds the poll will be active after creation, 5-600. Can't be used together with close date.

# close\_date: datetime.datetime | datetime.timedelta | int | 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: bool | None

Pass True if the poll needs to be immediately closed. This can be useful for poll preview.

#### disable\_notification: bool | None

Sends the message silently. Users will receive a notification with no sound.

## protect\_content: bool | Default | None

Protects the contents of the sent message from forwarding and saving

#### message\_effect\_id: str | None

Unique identifier of the message effect to be added to the message; for private chats only

#### reply\_parameters: ReplyParameters | None

Description of the message to reply to

# reply\_markup: InlineKeyboardMarkup | ReplyKeyboardMarkup | ReplyKeyboardRemove | ForceReply | None

Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

```
allow_sending_without_reply: bool | None
```

Pass True if the message should be sent even if the specified replied-to message is not found

Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023

```
reply_to_message_id: int | None
```

If the message is a reply, ID of the original message

Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023

# **Usage**

#### As bot method

```
result: Message = await bot.send_poll(...)
```

# Method as object

# Imports:

- from aiogram.methods.send\_poll import SendPoll
- alias: from aiogram.methods import SendPoll

## With specific bot

```
result: Message = await bot(SendPoll(...))
```

#### As reply into Webhook in handler

```
return SendPoll(...)
```

#### As shortcut from received object

- aiogram.types.message.Message.answer\_poll()
- aiogram.types.message.Message.reply\_poll()
- aiogram.types.chat\_join\_request.ChatJoinRequest.answer\_poll()
- aiogram.types.chat\_join\_request.ChatJoinRequest.answer\_poll\_pm()
- aiogram.types.chat\_member\_updated.ChatMemberUpdated.answer\_poll()
- aiogram.types.inaccessible\_message.InaccessibleMessage.answer\_poll()
- aiogram.types.inaccessible\_message.InaccessibleMessage.reply\_poll()

#### sendVenue

```
Returns: Message
```

```
class aiogram.methods.send_venue.SendVenue(*, chat_id: int | str, latitude: float, longitude: float, title: str,
                                                     address: str, business\_connection\_id: str \mid None = None,
                                                     message_thread_id: int | None = None, foursquare_id: str |
                                                     None = None, foursquare\_type: str | None = None,
                                                     google_place_id: str | None = None, google_place_type: str
                                                     | None = None, disable notification: bool | None = None,
                                                     protect_content: bool | ~aiogram.client.default.Default |
                                                     None = <Default('protect_content')>, message_effect_id: str
                                                     | None = None, reply\_parameters:
                                                     ~aiogram.types.reply parameters.ReplyParameters | None =
                                                     None, reply markup:
                                                     ~aiogram.types.inline_keyboard_markup.InlineKeyboardMarkup
                                                     ~aiogram.types.reply_keyboard_markup.ReplyKeyboardMarkup
                                                     ~aiogram.types.reply_keyboard_remove.ReplyKeyboardRemove
                                                     | ~aiogram.types.force_reply.ForceReply | None = None,
                                                     allow\_sending\_without\_reply: bool | None = None,
                                                     reply_to_message_id: int | None = None, **extra_data:
                                                     ~typing.Any)
```

Use this method to send information about a venue. On success, the sent aiogram.types.message.Message is returned.

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

#### chat\_id: int | str

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

## latitude: float

Latitude of the venue

# longitude: float

Longitude of the venue

#### title: str

Name of the venue

#### address: str

Address of the venue

#### business\_connection\_id: str | None

Unique identifier of the business connection on behalf of which the message will be sent

#### message\_thread\_id: int | None

Unique identifier for the target message thread (topic) of the forum; for forum supergroups only

#### foursquare\_id: str | None

Foursquare identifier of the venue

#### foursquare\_type: str | None

Foursquare type of the venue, if known. (For example, 'arts\_entertainment/default', 'arts\_entertainment/aquarium' or 'food/icecream'.)

## model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### google\_place\_id: str | None

Google Places identifier of the venue

# google\_place\_type: str | None

Google Places type of the venue. (See supported types.)

## disable\_notification: bool | None

Sends the message silently. Users will receive a notification with no sound.

#### protect\_content: bool | Default | None

Protects the contents of the sent message from forwarding and saving

## message\_effect\_id: str | None

Unique identifier of the message effect to be added to the message; for private chats only

## reply\_parameters: ReplyParameters | None

Description of the message to reply to

```
reply_markup: InlineKeyboardMarkup | ReplyKeyboardMarkup | ReplyKeyboardRemove |
ForceReply | None
```

Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

#### allow\_sending\_without\_reply: bool | None

Pass True if the message should be sent even if the specified replied-to message is not found

Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023

# reply\_to\_message\_id: int | None

If the message is a reply, ID of the original message

Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023

#### **Usage**

#### As bot method

```
result: Message = await bot.send_venue(...)
```

#### Method as object

#### Imports:

- from aiogram.methods.send\_venue import SendVenue
- alias: from aiogram.methods import SendVenue

# With specific bot

```
result: Message = await bot(SendVenue(...))
```

# As reply into Webhook in handler

```
return SendVenue(...)
```

# As shortcut from received object

- aiogram.types.message.Message.answer\_venue()
- aiogram.types.message.Message.reply\_venue()
- aiogram.types.chat\_join\_request.ChatJoinRequest.answer\_venue()
- aiogram.types.chat\_join\_request.ChatJoinRequest.answer\_venue\_pm()
- aiogram.types.chat\_member\_updated.ChatMemberUpdated.answer\_venue()
- aiogram.types.inaccessible\_message.InaccessibleMessage.answer\_venue()
- aiogram.types.inaccessible\_message.InaccessibleMessage.reply\_venue()

# sendVideo

Returns: Message

**class** aiogram.methods.send\_video.**SendVideo**(\*, chat\_id: int | str, video:

```
~aiogram.types.input_file.InputFile | str,
business connection id: str \mid None = None,
message_thread_id: int | None = None, duration: int | None
= None, width: int | None = None, height: int | None =
None, thumbnail: ~aiogram.types.input file.InputFile | None
= None, caption: str \mid None = None, parse mode: str \mid
~aiogram.client.default.Default | None =
<Default('parse mode')>, caption entities:
~typing.List[~aiogram.types.message_entity.MessageEntity]
| None = None, show\_caption\_above\_media: bool |
~aiogram.client.default.Default | None =
<Default('show_caption_above_media')>, has_spoiler: bool
| None = None, supports\_streaming: bool | None = None,
disable_notification: bool | None = None, protect_content:
bool | ~aiogram.client.default.Default | None =
<Default('protect_content')>, message_effect_id: str | None
= None, reply parameters:
~aiogram.types.reply_parameters.ReplyParameters | None =
None, reply markup:
~aiogram.types.inline_keyboard_markup.InlineKeyboardMarkup
~aiogram.types.reply_keyboard_markup.ReplyKeyboardMarkup
~aiogram.types.reply_keyboard_remove.ReplyKeyboardRemove
| \sim aiogram.types.force reply.ForceReply | None = None,
allow\_sending\_without\_reply: bool | None = None,
reply_to_message_id: int | None = None, **extra_data:
~typing.Any)
```

Use this method to send video files, Telegram clients support MPEG4 videos (other formats may be sent as aiogram.types.document.Document). On success, the sent aiogram.types.message.Message is returned. Bots can currently send video files of up to 50 MB in size, this limit may be changed in the future.

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

#### chat\_id: int | str

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

## video: InputFile | str

Video to send. Pass a file\_id as String to send a video that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a video from the Internet, or upload a new video using multipart/form-data. *More information on Sending Files* »

#### business\_connection\_id: str | None

Unique identifier of the business connection on behalf of which the message will be sent

#### message\_thread\_id: int | None

Unique identifier for the target message thread (topic) of the forum; for forum supergroups only

#### duration: int | None

Duration of sent video in seconds

#### width: int | None

Video width

## height: int | None

Video height

## thumbnail: InputFile | None

Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file\_attach\_name>' if the thumbnail was uploaded using multipart/form-data under <file attach name>. *More information on Sending Files* »

#### caption: str | None

Video caption (may also be used when resending videos by file\_id), 0-1024 characters after entities parsing

```
parse_mode: str | Default | None
```

Mode for parsing entities in the video caption. See formatting options for more details.

```
caption_entities: List[MessageEntity] | None
```

A JSON-serialized list of special entities that appear in the caption, which can be specified instead of parse\_mode

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
show_caption_above_media: bool | Default | None
```

Pass True, if the caption must be shown above the message media

```
has_spoiler: bool | None
```

Pass True if the video needs to be covered with a spoiler animation

```
supports_streaming: bool | None
```

Pass True if the uploaded video is suitable for streaming

```
disable_notification: bool | None
```

Sends the message silently. Users will receive a notification with no sound.

```
protect_content: bool | Default | None
```

Protects the contents of the sent message from forwarding and saving

```
message_effect_id: str | None
```

Unique identifier of the message effect to be added to the message; for private chats only

```
reply_parameters: ReplyParameters | None
```

Description of the message to reply to

```
reply_markup: InlineKeyboardMarkup | ReplyKeyboardMarkup | ReplyKeyboardRemove |
ForceReply | None
```

Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

```
allow_sending_without_reply: bool | None
```

Pass True if the message should be sent even if the specified replied-to message is not found

Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023

## reply\_to\_message\_id: int | None

If the message is a reply, ID of the original message

Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023

# **Usage**

#### As bot method

```
result: Message = await bot.send_video(...)
```

# Method as object

# Imports:

- from aiogram.methods.send\_video import SendVideo
- alias: from aiogram.methods import SendVideo

# With specific bot

```
result: Message = await bot(SendVideo(...))
```

#### As reply into Webhook in handler

```
return SendVideo(...)
```

## As shortcut from received object

- aiogram.types.message.Message.answer\_video()
- aiogram.types.message.Message.reply\_video()
- aiogram.types.chat\_join\_request.ChatJoinRequest.answer\_video()
- aiogram.types.chat\_join\_request.ChatJoinRequest.answer\_video\_pm()
- aiogram.types.chat\_member\_updated.ChatMemberUpdated.answer\_video()
- aiogram.types.inaccessible\_message.InaccessibleMessage.answer\_video()
- aiogram.types.inaccessible\_message.InaccessibleMessage.reply\_video()

#### sendVideoNote

Returns: Message

**class** aiogram.methods.send\_video\_note.**SendVideoNote**(\*, chat\_id: int | str, video\_note:

```
~aiogram.types.input_file.InputFile | str,
business\_connection\_id: str \mid None = None,
message\_thread\_id: int \mid None = None, duration:
int \mid None = None, length: int \mid None = None,
thumbnail: ~aiogram.types.input file.InputFile |
None = None, disable_notification: bool | None
= None, protect content: bool
~aiogram.client.default.Default | None =
< Default('protect content')>,
message effect id: str \mid None = None,
reply_parameters:
~aiogram.types.reply_parameters.ReplyParameters
| None = None, reply_markup:
~aiogram.types.inline_keyboard_markup.InlineKeyboardMarkup
~aiogram.types.reply_keyboard_markup.ReplyKeyboardMarkup
~aiogram.types.reply_keyboard_remove.ReplyKeyboardRemove
| ~aiogram.types.force_reply.ForceReply | None
= None, allow_sending_without_reply: bool |
None = None, reply to message id: int | None
= None, **extra data: ~typing.Any)
```

As of v.4.0, Telegram clients support rounded square MPEG4 videos of up to 1 minute long. Use this method to send video messages. On success, the sent aiogram. types.message is returned.

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

#### chat\_id: int | str

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

# video\_note: InputFile | str

Video note to send. Pass a file\_id as String to send a video note that exists on the Telegram servers (recommended) or upload a new video using multipart/form-data. *More information on Sending Files* ». Sending video notes by a URL is currently unsupported

#### business\_connection\_id: str | None

Unique identifier of the business connection on behalf of which the message will be sent

## message\_thread\_id: int | None

Unique identifier for the target message thread (topic) of the forum; for forum supergroups only

#### duration: int | None

Duration of sent video in seconds

## length: int | None

Video width and height, i.e. diameter of the video message

#### thumbnail: InputFile | None

Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused

and can be only uploaded as a new file, so you can pass 'attach://<file\_attach\_name>' if the thumbnail was uploaded using multipart/form-data under <file\_attach\_name>. *More information on Sending Files* »

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
disable_notification: bool | None
```

Sends the message silently. Users will receive a notification with no sound.

```
protect_content: bool | Default | None
```

Protects the contents of the sent message from forwarding and saving

```
message_effect_id: str | None
```

Unique identifier of the message effect to be added to the message; for private chats only

```
reply_parameters: ReplyParameters | None
```

Description of the message to reply to

```
reply_markup: InlineKeyboardMarkup | ReplyKeyboardMarkup | ReplyKeyboardRemove |
ForceReply | None
```

Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

```
allow_sending_without_reply: bool | None
```

Pass True if the message should be sent even if the specified replied-to message is not found

Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023

```
reply_to_message_id: int | None
```

If the message is a reply, ID of the original message

Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023

#### **Usage**

#### As bot method

```
result: Message = await bot.send_video_note(...)
```

# Method as object

# Imports:

- from aiogram.methods.send\_video\_note import SendVideoNote
- alias: from aiogram.methods import SendVideoNote

## With specific bot

```
result: Message = await bot(SendVideoNote(...))
```

# As reply into Webhook in handler

```
return SendVideoNote(...)
```

## As shortcut from received object

- aiogram.types.message.Message.answer\_video\_note()
- aiogram.types.message.Message.reply\_video\_note()
- aiogram.types.chat\_join\_request.ChatJoinRequest.answer\_video\_note()
- aiogram.types.chat\_join\_request.ChatJoinRequest.answer\_video\_note\_pm()
- aiogram.types.chat\_member\_updated.ChatMemberUpdated.answer\_video\_note()
- aiogram.types.inaccessible\_message.InaccessibleMessage.answer\_video\_note()
- aiogram.types.inaccessible\_message.InaccessibleMessage.reply\_video\_note()

#### sendVoice

```
Returns: Message
```

class aiogram.methods.send\_voice.SendVoice(\*, chat\_id: int | str, voice:

~aiogram.types.input\_file.InputFile | str, business\_connection\_id: str | None = None, message thread id: int | None = None, caption: str | None = None, parse\_mode: str | ~aiogram.client.default.Default |  $None = \langle Default('parse\_mode') \rangle$ , caption\_entities: ~typing.List[~aiogram.types.message\_entity.MessageEntity] | None = None, duration: int | None = None,*disable\_notification: bool* | *None* = *None*, *protect\_content:* bool | ~aiogram.client.default.Default | None = <Default('protect\_content')>, message\_effect\_id: str | None = None, reply\_parameters: ~aiogram.types.reply\_parameters.ReplyParameters | None = *None*, reply markup: ~aiogram.types.inline\_keyboard\_markup.InlineKeyboardMarkup ~aiogram.types.reply\_keyboard\_markup.ReplyKeyboardMarkup ~aiogram.types.reply keyboard remove.ReplyKeyboardRemove  $| \sim aiogram.types.force\_reply.ForceReply | None = None,$ 

allow\_sending\_without\_reply: bool | None = None, reply\_to\_message\_id: int | None = None, \*\*extra\_data:

2.3. Bot API 481

~typing.Any)

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, or in .MP3 format, or in .M4A format (other formats may be sent as <code>aiogram.types.audio.Audio</code> or <code>aiogram.types.document.Document</code>). On success, the sent <code>aiogram.types.message</code>. <code>Message</code> is returned. Bots can currently send voice messages of up to 50 MB in size, this limit may be changed in the future.

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

```
chat_id: int | str
```

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

```
voice: InputFile | str
```

Audio file to send. Pass a file\_id as String to send a file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a file from the Internet, or upload a new one using multipart/form-data. *More information on Sending Files* »

```
business_connection_id: str | None
```

Unique identifier of the business connection on behalf of which the message will be sent

```
message_thread_id: int | None
```

Unique identifier for the target message thread (topic) of the forum; for forum supergroups only

```
caption: str | None
```

Voice message caption, 0-1024 characters after entities parsing

```
parse_mode: str | Default | None
```

Mode for parsing entities in the voice message caption. See formatting options for more details.

```
caption_entities: List[MessageEntity] | None
```

A JSON-serialized list of special entities that appear in the caption, which can be specified instead of parse\_mode

```
duration: int | None
```

Duration of the voice message in seconds

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
disable_notification: bool | None
```

Sends the message silently. Users will receive a notification with no sound.

```
protect_content: bool | Default | None
```

Protects the contents of the sent message from forwarding and saving

```
message_effect_id: str | None
```

Unique identifier of the message effect to be added to the message; for private chats only

```
reply_parameters: ReplyParameters | None
```

Description of the message to reply to

```
reply_markup: InlineKeyboardMarkup | ReplyKeyboardMarkup | ReplyKeyboardRemove |
ForceReply | None
```

Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

## allow\_sending\_without\_reply: bool | None

Pass True if the message should be sent even if the specified replied-to message is not found

Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023

```
reply_to_message_id: int | None
```

If the message is a reply, ID of the original message

Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023

# **Usage**

#### As bot method

```
result: Message = await bot.send_voice(...)
```

# Method as object

# Imports:

- from aiogram.methods.send\_voice import SendVoice
- alias: from aiogram.methods import SendVoice

# With specific bot

```
result: Message = await bot(SendVoice(...))
```

# As reply into Webhook in handler

```
return SendVoice(...)
```

# As shortcut from received object

- aiogram.types.message.Message.answer\_voice()
- aiogram.types.message.Message.reply\_voice()
- aiogram.types.chat\_join\_request.ChatJoinRequest.answer\_voice()
- aiogram.types.chat\_join\_request.ChatJoinRequest.answer\_voice\_pm()
- aiogram.types.chat\_member\_updated.ChatMemberUpdated.answer\_voice()
- aiogram.types.inaccessible\_message.InaccessibleMessage.answer\_voice()
- aiogram.types.inaccessible\_message.InaccessibleMessage.reply\_voice()

#### setChatAdministratorCustomTitle

Returns: bool

 ${\bf class}\ aiogram.methods.set\_chat\_administrator\_custom\_title. {\bf SetChatAdministratorCustomTitle}(*,$ 

chat\_id:
int
|
str,
user\_id:
int,
custom\_title:
str,
\*\*extra\_data:
Any)

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

```
chat_id: int | str
```

Unique identifier for the target chat or username of the target supergroup (in the format @supergroupusername)

```
user_id: int
```

Unique identifier of the target user

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
custom title: str
```

New custom title for the administrator; 0-16 characters, emoji are not allowed

# **Usage**

#### As bot method

```
result: bool = await bot.set_chat_administrator_custom_title(...)
```

## Method as object

#### Imports:

- from aiogram.methods.set\_chat\_administrator\_custom\_title import SetChatAdministratorCustomTitle
- alias: from aiogram.methods import SetChatAdministratorCustomTitle

# With specific bot

```
result: bool = await bot(SetChatAdministratorCustomTitle(...))
```

## As reply into Webhook in handler

```
return SetChatAdministratorCustomTitle(...)
```

# As shortcut from received object

aiogram.types.chat.Chat.set\_administrator\_custom\_title()

# setChatDescription

Returns: bool

Use this method to change the description of a group, a supergroup or a channel. The bot must be an administrator in the chat for this to work and must have the appropriate administrator rights. Returns True on success.

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

```
chat_id: int | str
```

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
description: str | None
```

New chat description, 0-255 characters

#### As bot method

```
result: bool = await bot.set_chat_description(...)
```

# Method as object

Imports:

- from aiogram.methods.set\_chat\_description import SetChatDescription
- alias: from aiogram.methods import SetChatDescription

## With specific bot

```
result: bool = await bot(SetChatDescription(...))
```

# As reply into Webhook in handler

```
return SetChatDescription(...)
```

## As shortcut from received object

• aiogram.types.chat.Chat.set\_description()

## setChatMenuButton

Returns: bool

MenuButtonCommands |
MenuButtonWebApp |
MenuButtonDefault | None = None,
\*\*extra\_data: Any)

Use this method to change the bot's menu button in a private chat, or the default menu button. Returns True on success.

Source: https://core.telegram.org/bots/api#setchatmenubutton

```
chat_id: int | None
```

Unique identifier for the target private chat. If not specified, default bot's menu button will be changed

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

menu\_button: MenuButtonCommands | MenuButtonWebApp | MenuButtonDefault | None

A JSON-serialized object for the bot's new menu button. Defaults to aiogram.types.

menu\_button\_default.MenuButtonDefault

# **Usage**

#### As bot method

```
result: bool = await bot.set_chat_menu_button(...)
```

## Method as object

Imports:

- from aiogram.methods.set\_chat\_menu\_button import SetChatMenuButton
- alias: from aiogram.methods import SetChatMenuButton

# With specific bot

```
result: bool = await bot(SetChatMenuButton(...))
```

#### As reply into Webhook in handler

```
return SetChatMenuButton(...)
```

#### setChatPermissions

Returns: bool

class aiogram.methods.set\_chat\_permissions.SetChatPermissions(\*, chat\_id: int

| str, permissions: ChatPermissions, use\_independent\_chat\_permissions: bool | None = None, \*\*extra\_data: Any)

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* administrator rights. Returns True on success.

Source: https://core.telegram.org/bots/api#setchatpermissions

```
chat_id: int | str
```

Unique identifier for the target chat or username of the target supergroup (in the format @supergroupusername)

#### permissions: ChatPermissions

A JSON-serialized object for new default chat permissions

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
use_independent_chat_permissions: bool | None
```

Pass True if chat permissions are set independently. Otherwise, the <code>can\_send\_other\_messages</code> and <code>can\_add\_web\_page\_previews</code> permissions will imply the <code>can\_send\_messages</code>, <code>can\_send\_audios</code>, <code>can\_send\_documents</code>, <code>can\_send\_photos</code>, <code>can\_send\_videos</code>, <code>can\_send\_video\_notes</code>, and <code>can\_send\_voice\_notes</code> permissions; the <code>can\_send\_polls</code> permission will imply the <code>can\_send\_messages</code> permission.

# **Usage**

#### As bot method

```
result: bool = await bot.set_chat_permissions(...)
```

## Method as object

## Imports:

- from aiogram.methods.set\_chat\_permissions import SetChatPermissions
- alias: from aiogram.methods import SetChatPermissions

#### With specific bot

```
result: bool = await bot(SetChatPermissions(...))
```

# As reply into Webhook in handler

```
return SetChatPermissions(...)
```

# As shortcut from received object

• aiogram.types.chat.Chat.set\_permissions()

#### setChatPhoto

Returns: bool

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

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

```
chat_id: int | str
```

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

photo: InputFile

New chat photo, uploaded using multipart/form-data

# **Usage**

#### As bot method

```
result: bool = await bot.set_chat_photo(...)
```

# Method as object

Imports:

- from aiogram.methods.set\_chat\_photo import SetChatPhoto
- alias: from aiogram.methods import SetChatPhoto

# With specific bot

```
result: bool = await bot(SetChatPhoto(...))
```

## As shortcut from received object

• aiogram.types.chat.Chat.set\_photo()

#### setChatStickerSet

Returns: bool

Use this method to set a new group sticker set for a supergroup. The bot must be an administrator in the chat for this to work and must have the appropriate administrator rights. Use the field <code>can\_set\_sticker\_set</code> optionally returned in <code>aiogram.methods.get\_chat.GetChat</code> requests to check if the bot can use this method. Returns True on success.

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

```
chat_id: int | str
```

Unique identifier for the target chat or username of the target supergroup (in the format @supergroupusername)

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
sticker_set_name: str
```

Name of the sticker set to be set as the group sticker set

# **Usage**

#### As bot method

```
result: bool = await bot.set_chat_sticker_set(...)
```

#### Method as object

Imports:

- from aiogram.methods.set\_chat\_sticker\_set import SetChatStickerSet
- alias: from aiogram.methods import SetChatStickerSet

# With specific bot

```
result: bool = await bot(SetChatStickerSet(...))
```

# As reply into Webhook in handler

```
return SetChatStickerSet(...)
```

## As shortcut from received object

• aiogram.types.chat.Chat.set\_sticker\_set()

#### setChatTitle

Returns: bool

```
class aiogram.methods.set_chat_title.SetChatTitle(*, chat_id: int | str, title: str, **extra_data: Any)
```

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 administrator rights. Returns True on success.

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

```
chat_id: int | str
```

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
title: str
```

New chat title, 1-128 characters

## **Usage**

#### As bot method

```
result: bool = await bot.set_chat_title(...)
```

## Method as object

## Imports:

- from aiogram.methods.set\_chat\_title import SetChatTitle
- alias: from aiogram.methods import SetChatTitle

## With specific bot

```
result: bool = await bot(SetChatTitle(...))
```

## As reply into Webhook in handler

```
return SetChatTitle(...)
```

# As shortcut from received object

• aiogram.types.chat.Chat.set\_title()

# setMessageReaction

Returns: bool

 $\textbf{class} \ \texttt{aiogram.methods.set\_message\_reaction.SetMessageReaction} (*, \textit{chat\_id: int} \mid \textit{str}, \textit{message\_id: int}) (*, \textit{chat\_id: int} \mid \textit{str}, \textit{int} \mid \textit{st$ 

int, reaction: List[ReactionTypeEmoji | ReactionTypeCustomEmoji | ReactionTypePaid] | None = None, is\_big: bool | None = None, \*\*extra\_data: Any)

Use this method to change the chosen reactions on a message. Service messages can't be reacted to. Automatically forwarded messages from a channel to its discussion group have the same available reactions as messages in the channel. Bots can't use paid reactions. Returns True on success.

Source: https://core.telegram.org/bots/api#setmessagereaction

```
chat_id: int | str
```

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

## message\_id: int

Identifier of the target message. If the message belongs to a media group, the reaction is set to the first non-deleted message in the group instead.

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
reaction: List[ReactionTypeEmoji | ReactionTypeCustomEmoji | ReactionTypePaid] |
None
```

A JSON-serialized list of reaction types to set on the message. Currently, as non-premium users, bots can set up to one reaction per message. A custom emoji reaction can be used if it is either already present on the message or explicitly allowed by chat administrators. Paid reactions can't be used by bots.

```
is_big: bool | None
```

Pass True to set the reaction with a big animation

## **Usage**

#### As bot method

```
result: bool = await bot.set_message_reaction(...)
```

# Method as object

# Imports:

- from aiogram.methods.set\_message\_reaction import SetMessageReaction
- alias: from aiogram.methods import SetMessageReaction

## With specific bot

```
result: bool = await bot(SetMessageReaction(...))
```

# As reply into Webhook in handler

```
return SetMessageReaction(...)
```

# As shortcut from received object

• aiogram.types.message.Message.react()

# setMyCommands

Returns: bool

class aiogram.methods.set\_my\_commands.SetMyCommands(\*, commands: List[BotCommand], scope:

BotCommandScopeDefault |
BotCommandScopeAllPrivateChats |
BotCommandScopeAllGroupChats |
BotCommandScopeAllChatAdministrators |
BotCommandScopeChat |
BotCommandScopeChatAdministrators |
BotCommandScopeChatAdministrators |
BotCommandScopeChatMember | None = None,
language\_code: str | None = None,
\*\*extra\_data: Any)

Use this method to change the list of the bot's commands. See this manual for more details about bot commands. Returns True on success.

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

#### commands: List[BotCommand]

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.

```
scope: BotCommandScopeDefault | BotCommandScopeAllPrivateChats |
BotCommandScopeAllGroupChats | BotCommandScopeAllChatAdministrators |
BotCommandScopeChat | BotCommandScopeChatAdministrators | BotCommandScopeChatMember | None
```

A JSON-serialized object, describing scope of users for which the commands are relevant. Defaults to aiogram.types.bot\_command\_scope\_default.BotCommandScopeDefault.

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
language_code: str | None
```

A two-letter ISO 639-1 language code. If empty, commands will be applied to all users from the given scope, for whose language there are no dedicated commands

#### **Usage**

# As bot method

```
result: bool = await bot.set_my_commands(...)
```

# Method as object

# Imports:

- from aiogram.methods.set\_my\_commands import SetMyCommands
- alias: from aiogram.methods import SetMyCommands

## With specific bot

```
result: bool = await bot(SetMyCommands(...))
```

## As reply into Webhook in handler

```
return SetMyCommands(...)
```

## setMyDefaultAdministratorRights

Returns: bool

class aiogram.methods.set\_my\_default\_administrator\_rights.SetMyDefaultAdministratorRights(\*,

rights:
ChatAdministratorRights

|
None
=
None,
for\_channels:
bool
|
None
=
None,
\*\*extra\_data:
Any)

Use this method to change the default administrator rights requested by the bot when it's added as an administrator to groups or channels. These rights will be suggested to users, but they are free to modify the list before adding the bot. Returns True on success.

Source: https://core.telegram.org/bots/api#setmydefaultadministratorrights

# rights: ChatAdministratorRights | None

A JSON-serialized object describing new default administrator rights. If not specified, the default administrator rights will be cleared.

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
for_channels: bool | None
```

Pass True to change the default administrator rights of the bot in channels. Otherwise, the default administrator rights of the bot for groups and supergroups will be changed.

#### As bot method

```
result: bool = await bot.set_my_default_administrator_rights(...)
```

# Method as object

Imports:

- from aiogram.methods.set\_my\_default\_administrator\_rights import SetMyDefaultAdministratorRights
- alias: from aiogram.methods import SetMyDefaultAdministratorRights

## With specific bot

```
result: bool = await bot(SetMyDefaultAdministratorRights(...))
```

## As reply into Webhook in handler

```
return SetMyDefaultAdministratorRights(...)
```

# setMyDescription

Returns: bool

Use this method to change the bot's description, which is shown in the chat with the bot if the chat is empty. Returns True on success.

Source: https://core.telegram.org/bots/api#setmydescription

```
description: str | None
```

New bot description; 0-512 characters. Pass an empty string to remove the dedicated description for the given language.

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
language_code: str | None
```

A two-letter ISO 639-1 language code. If empty, the description will be applied to all users for whose language there is no dedicated description.

#### As bot method

```
result: bool = await bot.set_my_description(...)
```

# Method as object

Imports:

- from aiogram.methods.set\_my\_description import SetMyDescription
- alias: from aiogram.methods import SetMyDescription

## With specific bot

```
result: bool = await bot(SetMyDescription(...))
```

# As reply into Webhook in handler

```
return SetMyDescription(...)
```

## setMyName

Returns: bool

Use this method to change the bot's name. Returns True on success.

Source: https://core.telegram.org/bots/api#setmyname

name: str | None

New bot name; 0-64 characters. Pass an empty string to remove the dedicated name for the given language.

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model post init method.

language\_code: str | None

A two-letter ISO 639-1 language code. If empty, the name will be shown to all users for whose language there is no dedicated name.

#### As bot method

```
result: bool = await bot.set_my_name(...)
```

# Method as object

Imports:

- from aiogram.methods.set\_my\_name import SetMyName
- alias: from aiogram.methods import SetMyName

## With specific bot

```
result: bool = await bot(SetMyName(...))
```

# As reply into Webhook in handler

```
return SetMyName(...)
```

## setMyShortDescription

Returns: bool

Use this method to change the bot's short description, which is shown on the bot's profile page and is sent together with the link when users share the bot. Returns True on success.

Source: https://core.telegram.org/bots/api#setmyshortdescription

```
short_description: str | None
```

New short description for the bot; 0-120 characters. Pass an empty string to remove the dedicated short description for the given language.

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
language_code: str | None
```

A two-letter ISO 639-1 language code. If empty, the short description will be applied to all users for whose language there is no dedicated short description.

#### Usage

#### As bot method

```
result: bool = await bot.set_my_short_description(...)
```

# Method as object

Imports:

- from aiogram.methods.set\_my\_short\_description import SetMyShortDescription
- alias: from aiogram.methods import SetMyShortDescription

### With specific bot

```
result: bool = await bot(SetMyShortDescription(...))
```

# As reply into Webhook in handler

```
return SetMyShortDescription(...)
```

#### unbanChatMember

Returns: bool

Use this method to unban a previously banned 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

```
chat_id: int | str
```

Unique identifier for the target group or username of the target supergroup or channel (in the format @channelusername)

user\_id: int

Unique identifier of the target user

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
only_if_banned: bool | None
```

Do nothing if the user is not banned

## **Usage**

#### As bot method

```
result: bool = await bot.unban_chat_member(...)
```

## Method as object

Imports:

- from aiogram.methods.unban\_chat\_member import UnbanChatMember
- alias: from aiogram.methods import UnbanChatMember

# With specific bot

```
result: bool = await bot(UnbanChatMember(...))
```

# As reply into Webhook in handler

```
return UnbanChatMember(...)
```

### As shortcut from received object

• aiogram.types.chat.Chat.unban()

## unbanChatSenderChat

Returns: bool

Use this method to unban a previously banned channel chat in a supergroup or channel. The bot must be an administrator for this to work and must have the appropriate administrator rights. Returns True on success.

Source: https://core.telegram.org/bots/api#unbanchatsenderchat

```
chat_id: int | str
```

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
sender_chat_id: int
```

Unique identifier of the target sender chat

#### **Usage**

#### As bot method

```
result: bool = await bot.unban_chat_sender_chat(...)
```

# Method as object

Imports:

- from aiogram.methods.unban\_chat\_sender\_chat import UnbanChatSenderChat
- alias: from aiogram.methods import UnbanChatSenderChat

# With specific bot

```
result: bool = await bot(UnbanChatSenderChat(...))
```

### As reply into Webhook in handler

```
return UnbanChatSenderChat(...)
```

# As shortcut from received object

• aiogram.types.chat.Chat.unban\_sender\_chat()

# unhideGeneralForumTopic

Returns: bool

Use this method to unhide the 'General' topic in a forum supergroup chat. The bot must be an administrator in the chat for this to work and must have the *can\_manage\_topics* administrator rights. Returns True on success.

Source: https://core.telegram.org/bots/api#unhidegeneralforumtopic

```
chat_id: int | str
```

Unique identifier for the target chat or username of the target supergroup (in the format @supergroupusername)

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### **Usage**

#### As bot method

```
result: bool = await bot.unhide_general_forum_topic(...)
```

### Method as object

Imports:

- from aiogram.methods.unhide\_general\_forum\_topic import UnhideGeneralForumTopic
- alias: from aiogram.methods import UnhideGeneralForumTopic

# With specific bot

```
result: bool = await bot(UnhideGeneralForumTopic(...))
```

### As reply into Webhook in handler

```
return UnhideGeneralForumTopic(...)
```

#### unpinAllChatMessages

Returns: bool

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' administrator right in a supergroup or 'can\_edit\_messages' administrator right in a channel. Returns True on success.

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

```
chat_id: int | str
```

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

# **Usage**

#### As bot method

```
result: bool = await bot.unpin_all_chat_messages(...)
```

# Method as object

Imports:

- from aiogram.methods.unpin\_all\_chat\_messages import UnpinAllChatMessages
- alias: from aiogram.methods import UnpinAllChatMessages

## With specific bot

```
result: bool = await bot(UnpinAllChatMessages(...))
```

# As reply into Webhook in handler

```
return UnpinAllChatMessages(...)
```

## As shortcut from received object

• aiogram.types.chat.Chat.unpin\_all\_messages()

# unpinAllForumTopicMessages

Returns: bool

```
{\bf class} \ \ {\bf aiogram.methods.unpin\_all\_forum\_topic\_messages. {\bf UnpinAllForumTopicMessages}(*, chat\_id:int \mid str, )
```

mu | str, message\_thread\_id: int, \*\*extra\_data: Any)

Use this method to clear the list of pinned messages in a forum topic. The bot must be an administrator in the chat for this to work and must have the *can\_pin\_messages* administrator right in the supergroup. Returns True on success.

Source: https://core.telegram.org/bots/api#unpinallforumtopicmessages

# chat\_id: int | str

Unique identifier for the target chat or username of the target supergroup (in the format @supergroupusername)

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
message_thread_id: int
```

Unique identifier for the target message thread of the forum topic

# **Usage**

#### As bot method

```
result: bool = await bot.unpin_all_forum_topic_messages(...)
```

# Method as object

Imports:

- from aiogram.methods.unpin\_all\_forum\_topic\_messages import UnpinAllForumTopicMessages
- alias: from aiogram.methods import UnpinAllForumTopicMessages

### With specific bot

```
result: bool = await bot(UnpinAllForumTopicMessages(...))
```

#### As reply into Webhook in handler

```
return UnpinAllForumTopicMessages(...)
```

# unpinAllGeneralForumTopicMessages

Returns: bool

```
\textbf{class} \ \texttt{aiogram.methods.unpin\_all\_general\_forum\_topic\_messages. \textbf{UnpinAllGeneralForumTopicMessages} (*, \texttt{aiogram.methods.unpin\_all\_general\_forum\_topic\_messages) (*, \texttt{aiogram.methods.unpin\_all\_general\_forum\_topic\_mes
```

```
chat_id:
int
|
str,
**ex-
tra_data:
Any)
```

Use this method to clear the list of pinned messages in a General forum topic. The bot must be an administrator in the chat for this to work and must have the *can\_pin\_messages* administrator right in the supergroup. Returns True on success.

Source: https://core.telegram.org/bots/api#unpinallgeneralforumtopicmessages

# chat\_id: int | str

Unique identifier for the target chat or username of the target supergroup (in the format @supergroupusername)

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model post init method.

#### **Usage**

#### As bot method

```
result: bool = await bot.unpin_all_general_forum_topic_messages(...)
```

### Method as object

Imports:

- from aiogram.methods.unpin\_all\_general\_forum\_topic\_messages import UnpinAllGeneralForumTopicMessages
- alias: from aiogram.methods import UnpinAllGeneralForumTopicMessages

# With specific bot

```
result: bool = await bot(UnpinAllGeneralForumTopicMessages(...))
```

# As reply into Webhook in handler

```
return UnpinAllGeneralForumTopicMessages(...)
```

#### As shortcut from received object

• aiogram.types.chat.Chat.unpin\_all\_general\_forum\_topic\_messages()

#### unpinChatMessage

Returns: bool

```
class aiogram.methods.unpin_chat_message.UnpinChatMessage(*, chat_id: int | str,
```

business\_connection\_id: str | None = None, message\_id: int | None = None, \*\*extra\_data: Any)

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' administrator right in a supergroup or 'can\_edit\_messages' administrator right in a channel. Returns True on success.

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

```
chat_id: int | str
```

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

```
business_connection_id: str | None
```

Unique identifier of the business connection on behalf of which the message will be unpinned

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
message_id: int | None
```

Identifier of the message to unpin. Required if *business\_connection\_id* is specified. If not specified, the most recent pinned message (by sending date) will be unpinned.

#### **Usage**

#### As bot method

```
result: bool = await bot.unpin_chat_message(...)
```

# Method as object

Imports:

- from aiogram.methods.unpin\_chat\_message import UnpinChatMessage
- alias: from aiogram.methods import UnpinChatMessage

# With specific bot

```
result: bool = await bot(UnpinChatMessage(...))
```

#### As reply into Webhook in handler

```
return UnpinChatMessage(...)
```

# As shortcut from received object

- aiogram.types.chat.Chat.unpin\_message()
- aiogram.types.message.Message.unpin()

## **Updating messages**

## deleteMessage

Returns: bool

Use this method to delete a message, including service messages, with the following limitations:

- A message can only be deleted if it was sent less than 48 hours ago.
- Service messages about a supergroup, channel, or forum topic creation can't be deleted.
- A dice message in a private chat can only be deleted if it was sent more than 24 hours ago.
- Bots can delete outgoing messages in private chats, groups, and supergroups.
- Bots can delete incoming messages in private chats.
- Bots granted *can\_post\_messages* permissions can delete outgoing messages in channels.
- If the bot is an administrator of a group, it can delete any message there.
- If the bot has can\_delete\_messages permission in a supergroup or a channel, it can delete any message there.

Returns True on success.

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

```
chat_id: int | str
```

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
message_id: int
```

Identifier of the message to delete

### **Usage**

#### As bot method

```
result: bool = await bot.delete_message(...)
```

# Method as object

#### Imports:

- from aiogram.methods.delete\_message import DeleteMessage
- alias: from aiogram.methods import DeleteMessage

### With specific bot

```
result: bool = await bot(DeleteMessage(...))
```

# As reply into Webhook in handler

```
return DeleteMessage(...)
```

# As shortcut from received object

- aiogram.types.chat.Chat.delete\_message()
- aiogram.types.message.Message.delete()

# deleteMessages

Returns: bool

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

Source: https://core.telegram.org/bots/api#deletemessages

```
chat_id: int | str
```

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
message_ids: List[int]
```

A JSON-serialized list of 1-100 identifiers of messages to delete. See aiogram.methods. delete\_message.DeleteMessage for limitations on which messages can be deleted

# **Usage**

#### As bot method

```
result: bool = await bot.delete_messages(...)
```

# Method as object

Imports:

- from aiogram.methods.delete\_messages import DeleteMessages
- alias: from aiogram.methods import DeleteMessages

# With specific bot

```
result: bool = await bot(DeleteMessages(...))
```

# As reply into Webhook in handler

```
return DeleteMessages(...)
```

### editMessageCaption

```
Returns: Union[Message, bool]
```

```
None = None, chat_id: int | str |
None = None, message_id: int |
None = None, inline_message_id:
str | None = None, caption: str |
None = None, parse_mode: str |
~aiogram.client.default.Default |
None = <Default('parse_mode')>,
caption_entities: ~typ-
ing.List[~aiogram.types.message_entity.MessageEntit]
| None = None,
show_caption_above_media: bool |
```

~aiogram.client.default.Default | None = <Default('show\_caption\_above\_media')>,

reply\_markup:

~aiogram.types.inline\_keyboard\_markup.InlineKeybo | None = None, \*\*extra\_data:

~typing.Any)

Use this method to edit captions of messages. On success, if the edited message is not an inline message, the

edited aiogram.types.message.Message is returned, otherwise True is returned. Note that business messages that were not sent by the bot and do not contain an inline keyboard can only be edited within 48 hours from the time they were sent.

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

```
business_connection_id: str | None
```

Unique identifier of the business connection on behalf of which the message to be edited was sent

```
chat_id: int | str | None
```

Required if *inline\_message\_id* is not specified. Unique identifier for the target chat or username of the target channel (in the format @channelusername)

```
message_id: int | None
```

Required if *inline\_message\_id* is not specified. Identifier of the message to edit

```
inline_message_id: str | None
```

Required if chat\_id and message\_id are not specified. Identifier of the inline message

```
caption: str | None
```

New caption of the message, 0-1024 characters after entities parsing

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
parse_mode: str | Default | None
```

Mode for parsing entities in the message caption. See formatting options for more details.

```
caption_entities: List[MessageEntity] | None
```

A JSON-serialized list of special entities that appear in the caption, which can be specified instead of parse\_mode

```
show_caption_above_media: bool | Default | None
```

Pass True, if the caption must be shown above the message media. Supported only for animation, photo and video messages.

```
reply_markup: InlineKeyboardMarkup | None
```

A JSON-serialized object for an inline keyboard.

# **Usage**

## As bot method

```
result: Union[Message, bool] = await bot.edit_message_caption(...)
```

# Method as object

## Imports:

- from aiogram.methods.edit\_message\_caption import EditMessageCaption
- alias: from aiogram.methods import EditMessageCaption

# With specific bot

```
result: Union[Message, bool] = await bot(EditMessageCaption(...))
```

# As reply into Webhook in handler

```
return EditMessageCaption(...)
```

# As shortcut from received object

• aiogram.types.message.Message.edit\_caption()

# editMessageLiveLocation

```
Returns: Union[Message, bool]
```

 $\textbf{class} \ \texttt{aiogram.methods.edit\_message\_live\_location.} \textbf{EditMessageLiveLocation} (*, \textit{latitude: float}, \textit{latitude: flo$ 

longitude: float, business\_connection\_id:  $str \mid None = None,$ chat\_id: int | str | None = None, message\_id: int | None = None,inline\_message\_id:  $str \mid None = None,$ live\_period: int | None = None,horizontal\_accuracy:  $float \mid None = None,$ heading: int | None = None, proximity\_alert\_radius: int | None = None,reply markup: InlineKeyboardMarkup | None = None,\*\*extra\_data: Any)

Use this method to edit live location messages. A location can be edited until its *live\_period* expires or editing is explicitly disabled by a call to <code>aiogram.methods.stop\_message\_live\_location</code>. StopMessageLiveLocation. On success, if the edited message is not an inline message, the edited <code>aiogram.types.message</code>. Message is returned, otherwise True is returned.

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

latitude: float

Latitude of new location

longitude: float

Longitude of new location

#### business\_connection\_id: str | None

Unique identifier of the business connection on behalf of which the message to be edited was sent

#### chat\_id: int | str | None

Required if *inline\_message\_id* is not specified. Unique identifier for the target chat or username of the target channel (in the format @channelusername)

#### message\_id: int | None

Required if *inline\_message\_id* is not specified. Identifier of the message to edit

#### inline\_message\_id: str | None

Required if chat\_id and message\_id are not specified. Identifier of the inline message

### model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

#### **model\_post\_init**( $context: Any, /) \rightarrow None$

We need to both initialize private attributes and call the user-defined model\_post\_init method.

# live\_period: int | None

New period in seconds during which the location can be updated, starting from the message send date. If 0x7FFFFFF is specified, then the location can be updated forever. Otherwise, the new value must not exceed the current *live\_period* by more than a day, and the live location expiration date must remain within the next 90 days. If not specified, then *live\_period* remains unchanged

# horizontal\_accuracy: float | None

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

#### heading: int | None

Direction in which the user is moving, in degrees. Must be between 1 and 360 if specified.

#### proximity\_alert\_radius: int | None

The maximum distance for proximity alerts about approaching another chat member, in meters. Must be between 1 and 100000 if specified.

### reply\_markup: InlineKeyboardMarkup | None

A JSON-serialized object for a new inline keyboard.

## Usage

#### As bot method

```
result: Union[Message, bool] = await bot.edit_message_live_location(...)
```

# Method as object

Imports:

- from aiogram.methods.edit\_message\_live\_location import EditMessageLiveLocation
- alias: from aiogram.methods import EditMessageLiveLocation

### With specific bot

```
result: Union[Message, bool] = await bot(EditMessageLiveLocation(...))
```

# As reply into Webhook in handler

```
return EditMessageLiveLocation(...)
```

### As shortcut from received object

• aiogram.types.message.Message.edit\_live\_location()

### editMessageMedia

```
Returns: Union[Message, bool]
```

 $\textbf{class} \ aiogram. \texttt{methods.edit\_message\_media.EditMessageMedia} (*, \textit{media}: InputMediaAnimation | \texttt{message\_media.EditMessageMedia})) (*, \textit{media}: InputMediaAnimation | \texttt{message\_media})) (*, \textit{message\_media})) (*, \textit{message\_media}) (*, \textit{message\_media}) (*, \textit{message\_media}) (*, \textit{message\_media})) (*, \textit{message\_media}) (*, \textit{message\_me$ 

InputMediaDocument | InputMediaAudio | InputMediaPhoto | InputMediaVideo,

business\_connection\_id: str | None =
None, chat\_id: int | str | None = None,
message\_id: int | None = None,
inline\_message\_id: str | None = None,
reply\_markup: InlineKeyboardMarkup |
None = None, \*\*extra\_data: Any)

Use this method to edit animation, audio, document, photo, or video messages. If a message is part of a message album, then it can be edited only to an audio for audio albums, only to a document for document albums and to a photo or a video otherwise. When an inline message is edited, a new file can't be uploaded; use a previously uploaded file via its file\_id or specify a URL. On success, if the edited message is not an inline message, the edited <code>aiogram.types.message.Message</code> is returned, otherwise <code>True</code> is returned. Note that business messages that were not sent by the bot and do not contain an inline keyboard can only be edited within <code>48 hours</code> from the time they were sent.

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

media: InputMediaAnimation | InputMediaDocument | InputMediaAudio | InputMediaPhoto
| InputMediaVideo

A JSON-serialized object for a new media content of the message

```
business_connection_id: str | None
```

Unique identifier of the business connection on behalf of which the message to be edited was sent

```
chat_id: int | str | None
```

Required if *inline\_message\_id* is not specified. Unique identifier for the target chat or username of the target channel (in the format @channelusername)

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
message_id: int | None
```

Required if *inline\_message\_id* is not specified. Identifier of the message to edit

```
inline_message_id: str | None
```

Required if *chat\_id* and *message\_id* are not specified. Identifier of the inline message

```
reply_markup: InlineKeyboardMarkup | None
```

A JSON-serialized object for a new inline keyboard.

## **Usage**

#### As bot method

```
result: Union[Message, bool] = await bot.edit_message_media(...)
```

#### Method as object

Imports:

- from aiogram.methods.edit\_message\_media import EditMessageMedia
- alias: from aiogram.methods import EditMessageMedia

#### With specific bot

```
result: Union[Message, bool] = await bot(EditMessageMedia(...))
```

### As reply into Webhook in handler

```
return EditMessageMedia(...)
```

# As shortcut from received object

• aiogram.types.message.Message.edit\_media()

# editMessageReplyMarkup

Returns: Union[Message, bool]

class aiogram.methods.edit\_message\_reply\_markup.EditMessageReplyMarkup(\*, busi-

ness\_connection\_id: str | None = None, chat\_id: int | str | None = None, message\_id: int | None = None, inline\_message\_id: str | None = None, reply\_markup: InlineKeyboardMarkup | None = None, \*\*extra\_data: Any)

Use this method to edit only the reply markup of messages. On success, if the edited message is not an inline message, the edited <code>aiogram.types.message.Message</code> is returned, otherwise True is returned. Note that business messages that were not sent by the bot and do not contain an inline keyboard can only be edited within <code>48 hours</code> from the time they were sent.

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

```
business_connection_id: str | None
```

Unique identifier of the business connection on behalf of which the message to be edited was sent

```
chat_id: int | str | None
```

Required if *inline\_message\_id* is not specified. Unique identifier for the target chat or username of the target channel (in the format @channelusername)

```
message_id: int | None
```

Required if inline\_message\_id is not specified. Identifier of the message to edit

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
inline_message_id: str | None
```

Required if chat\_id and message\_id are not specified. Identifier of the inline message

reply\_markup: InlineKeyboardMarkup | None

A JSON-serialized object for an inline keyboard.

# **Usage**

#### As bot method

```
result: Union[Message, bool] = await bot.edit_message_reply_markup(...)
```

# Method as object

# Imports:

- from aiogram.methods.edit\_message\_reply\_markup import EditMessageReplyMarkup
- alias: from aiogram.methods import EditMessageReplyMarkup

# With specific bot

```
result: Union[Message, bool] = await bot(EditMessageReplyMarkup(...))
```

# As reply into Webhook in handler

```
return EditMessageReplyMarkup(...)
```

# As shortcut from received object

- aiogram.types.message.Message.edit\_reply\_markup()
- aiogram.types.message.Message.delete\_reply\_markup()

## editMessageText

Returns: Union[Message, bool]

```
class aiogram.methods.edit_message_text.EditMessageText(*.text: str, business connection id: str|
                                                                     None = None, chat id: int | str | None =
                                                                     None, message id: int | None = None,
                                                                     inline_message_id: str | None = None,
                                                                     parse mode: str |
                                                                      ~aiogram.client.default.Default | None =
                                                                      <Default('parse mode')>, entities: ~typ-
                                                                     ing.List[~aiogram.types.message_entity.MessageEntity]
                                                                     | None = None, link_preview_options:
                                                                      ~aiogram.types.link_preview_options.LinkPreviewOptions
                                                                     | ~aiogram.client.default.Default | None =
                                                                      <Default('link_preview')>, reply_markup:
                                                                     ~aiogram.types.inline_keyboard_markup.InlineKeyboardMark
                                                                     | None = None,
                                                                     disable_web_page_preview: bool |
                                                                      ~aiogram.client.default.Default | None =
                                                                      < Default('link_preview_is_disabled')>,
                                                                      **extra data: ~typing.Any)
```

Use this method to edit text and game messages. On success, if the edited message is not an inline message, the edited <code>aiogram.types.message.Message</code> is returned, otherwise <code>True</code> is returned. Note that business messages that were not sent by the bot and do not contain an inline keyboard can only be edited within <code>48 hours</code> from the time they were sent.

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

#### text: str

New text of the message, 1-4096 characters after entities parsing

#### business\_connection\_id: str | None

Unique identifier of the business connection on behalf of which the message to be edited was sent

# chat\_id: int | str | None

Required if *inline\_message\_id* is not specified. Unique identifier for the target chat or username of the target channel (in the format @channelusername)

#### message\_id: int | None

Required if *inline\_message\_id* is not specified. Identifier of the message to edit

#### inline\_message\_id: str | None

Required if chat\_id and message\_id are not specified. Identifier of the inline message

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
parse_mode: str | Default | None
```

Mode for parsing entities in the message text. See formatting options for more details.

```
entities: List[MessageEntity] | None
```

A JSON-serialized list of special entities that appear in message text, which can be specified instead of parse\_mode

link\_preview\_options: LinkPreviewOptions | Default | None

Link preview generation options for the message

```
reply_markup: InlineKeyboardMarkup | None
```

A JSON-serialized object for an inline keyboard.

```
disable_web_page_preview: bool | Default | None
```

Disables link previews for links in this message

Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023

# **Usage**

#### As bot method

```
result: Union[Message, bool] = await bot.edit_message_text(...)
```

## Method as object

# Imports:

- from aiogram.methods.edit\_message\_text import EditMessageText
- alias: from aiogram.methods import EditMessageText

## With specific bot

```
result: Union[Message, bool] = await bot(EditMessageText(...))
```

# As reply into Webhook in handler

```
return EditMessageText(...)
```

# As shortcut from received object

• aiogram.types.message.Message.edit\_text()

# stopMessageLiveLocation

Returns: Union[Message, bool]

class aiogram.methods.stop\_message\_live\_location.StopMessageLiveLocation(\*, busi-

ness\_connection\_id: str | None = None, chat\_id: int | str | None = None, message\_id: int | None = None, inline\_message\_id: str | None = None, reply\_markup: InlineKeyboardMarkup | None = None, \*\*extra data: Any)

Use this method to stop updating a live location message before *live\_period* expires. On success, if the message is not an inline message, the edited *aiogram.types.message*. Message is returned, otherwise True is returned.

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

```
business_connection_id: str | None
```

Unique identifier of the business connection on behalf of which the message to be edited was sent

```
chat_id: int | str | None
```

Required if *inline\_message\_id* is not specified. Unique identifier for the target chat or username of the target channel (in the format @channelusername)

```
message_id: int | None
```

Required if inline\_message\_id is not specified. Identifier of the message with live location to stop

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
inline_message_id: str | None
```

Required if chat\_id and message\_id are not specified. Identifier of the inline message

```
reply_markup: InlineKeyboardMarkup | None
```

A JSON-serialized object for a new inline keyboard.

#### **Usage**

#### As bot method

```
result: Union[Message, bool] = await bot.stop_message_live_location(...)
```

# Method as object

Imports:

- from aiogram.methods.stop\_message\_live\_location import StopMessageLiveLocation
- alias: from aiogram.methods import StopMessageLiveLocation

## With specific bot

```
result: Union[Message, bool] = await bot(StopMessageLiveLocation(...))
```

# As reply into Webhook in handler

```
return StopMessageLiveLocation(...)
```

# As shortcut from received object

• aiogram.types.message.Message.stop\_live\_location()

# stopPoll

```
Returns: Poll
```

```
class aiogram.methods.stop_poll.StopPoll(*, chat_id: int | str, message_id: int, business_connection_id: str | None = None, reply_markup: InlineKeyboardMarkup | None = None, **extra_data: Any)
```

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

Source: https://core.telegram.org/bots/api#stoppoll

```
chat_id: int | str
```

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

```
message_id: int
```

Identifier of the original message with the poll

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
business_connection_id: str | None
```

Unique identifier of the business connection on behalf of which the message to be edited was sent

```
reply_markup: InlineKeyboardMarkup | None
```

A JSON-serialized object for a new message inline keyboard.

# Usage

# As bot method

```
result: Poll = await bot.stop_poll(...)
```

# Method as object

Imports:

- from aiogram.methods.stop\_poll import StopPoll
- alias: from aiogram.methods import StopPoll

# With specific bot

```
result: Poll = await bot(StopPoll(...))
```

# As reply into Webhook in handler

```
return StopPoll(...)
```

# Inline mode

# answerInlineQuery

Returns: bool

class aiogram.methods.answer\_inline\_query.AnswerInlineQuery(\*, inline\_query\_id: str, results:

*List*/InlineQueryResultCachedAudio InlineQueryResultCachedDocument | InlineQueryResultCachedGif | InlineQueryResultCachedMpeg4Gif | InlineQueryResultCachedPhoto | InlineOuervResultCachedSticker | InlineQueryResultCachedVideo | InlineQueryResultCachedVoice | InlineQueryResultArticle | InlineQueryResultAudio | InlineQueryResultContact | InlineOueryResultGame | InlineQueryResultDocument | InlineQueryResultGif | InlineQueryResultLocation | InlineQueryResultMpeg4Gif | InlineOueryResultPhoto | InlineQueryResultVenue | InlineQueryResultVideo | InlineQueryResultVoice], cache\_time:  $int \mid None = None, is personal: bool \mid$  $None = None, next\_offset: str | None$ = None, button: InlineQueryResultsButton | None = None, switch pm parameter: str  $None = None, switch\_pm\_text: str$  $None = None, **extra_data: Any)$ 

Use this method to send answers to an inline query. On success, True is returned.

No more than **50** results per query are allowed.

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

#### inline\_query\_id: str

Unique identifier for the answered query

```
results: List[InlineQueryResultCachedAudio | InlineQueryResultCachedDocument |
InlineQueryResultCachedGif | InlineQueryResultCachedMpeg4Gif |
InlineQueryResultCachedPhoto | InlineQueryResultCachedSticker |
InlineQueryResultCachedVideo | InlineQueryResultCachedVoice |
InlineQueryResultArticle | InlineQueryResultAudio | InlineQueryResultContact |
InlineQueryResultGame | InlineQueryResultDocument | InlineQueryResultGif |
InlineQueryResultLocation | InlineQueryResultMpeg4Gif | InlineQueryResultPhoto |
InlineQueryResultVenue | InlineQueryResultVideo | InlineQueryResultVoice]
```

A JSON-serialized array of results for the inline query

#### cache\_time: int | None

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: bool | None

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.

### model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
next_offset: str | None
```

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.

### button: InlineQueryResultsButton | None

A JSON-serialized object describing a button to be shown above inline query results

```
switch_pm_parameter: str | None
```

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.

Deprecated since version API:6.7: https://core.telegram.org/bots/api-changelog#april-21-2023

```
switch_pm_text: str | None
```

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* 

Deprecated since version API:6.7: https://core.telegram.org/bots/api-changelog#april-21-2023

# **Usage**

# As bot method

```
result: bool = await bot.answer_inline_query(...)
```

#### Method as object

#### Imports:

- from aiogram.methods.answer\_inline\_query import AnswerInlineQuery
- alias: from aiogram.methods import AnswerInlineQuery

# With specific bot

```
result: bool = await bot(AnswerInlineQuery(...))
```

# As reply into Webhook in handler

```
return AnswerInlineQuery(...)
```

# As shortcut from received object

• aiogram.types.inline\_query.InlineQuery.answer()

### answerWebAppQuery

Returns: SentWebAppMessage

 $\textbf{class} \ \, \textbf{aiogram.methods.answer\_web\_app\_query.AnswerWebAppQuery} (*, \textit{web\_app\_query\_id: str, result: }) \\$ 

InlineQueryResultCachedAudio | InlineQueryResultCachedDocument | InlineQueryResultCachedGif | InlineQueryResultCachedMpeg4Gif | InlineQueryResultCachedPhoto | InlineQueryResultCachedSticker | InlineOueryResultCachedVideo | InlineQueryResultCachedVoice | InlineQueryResultArticle InlineQueryResultAudio | InlineQueryResultContact | InlineQueryResultGame | InlineQueryResultDocument | InlineQueryResultGif | InlineQueryResultLocation | InlineQueryResultMpeg4Gif | InlineQueryResultPhoto | InlineQueryResultVenue | InlineQueryResultVideo | InlineQueryResultVoice, \*\*extra\_data: Any)

Use this method to set the result of an interaction with a Web App and send a corresponding message on behalf of the user to the chat from which the query originated. On success, a <code>aiogram.types.sent\_web\_app\_message.SentWebAppMessage</code> object is returned.

Source: https://core.telegram.org/bots/api#answerwebappquery

```
web_app_query_id: str
```

Unique identifier for the query to be answered

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
result: InlineQueryResultCachedAudio | InlineQueryResultCachedDocument |
InlineQueryResultCachedGif | InlineQueryResultCachedMpeg4Gif |
InlineQueryResultCachedPhoto | InlineQueryResultCachedSticker |
InlineQueryResultCachedVideo | InlineQueryResultCachedVoice |
InlineQueryResultArticle | InlineQueryResultAudio | InlineQueryResultContact |
InlineQueryResultGame | InlineQueryResultDocument | InlineQueryResultGif |
InlineQueryResultLocation | InlineQueryResultMpeg4Gif | InlineQueryResultPhoto |
InlineQueryResultVenue | InlineQueryResultVideo | InlineQueryResultVoice
```

A JSON-serialized object describing the message to be sent

## **Usage**

#### As bot method

```
result: SentWebAppMessage = await bot.answer_web_app_query(...)
```

### Method as object

## Imports:

- from aiogram.methods.answer\_web\_app\_query import AnswerWebAppQuery
- alias: from aiogram.methods import AnswerWebAppQuery

#### With specific bot

```
result: SentWebAppMessage = await bot(AnswerWebAppQuery(...))
```

## As reply into Webhook in handler

```
return AnswerWebAppQuery(...)
```

# Games

## getGameHighScores

```
Returns: List[GameHighScore]
```

Use this method to get data for high score tables. Will return the score of the specified user and several of their neighbors in a game. Returns an Array of aiogram.types.game\_high\_score.GameHighScore objects.

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

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

user\_id: int

Target user id

chat\_id: int | None

Required if *inline\_message\_id* is not specified. Unique identifier for the target chat

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
message_id: int | None
```

Required if *inline\_message\_id* is not specified. Identifier of the sent message

```
inline_message_id: str | None
```

Required if *chat\_id* and *message\_id* are not specified. Identifier of the inline message

# **Usage**

#### As bot method

```
result: List[GameHighScore] = await bot.get_game_high_scores(...)
```

#### Method as object

Imports:

- from aiogram.methods.get\_game\_high\_scores import GetGameHighScores
- alias: from aiogram.methods import GetGameHighScores

# With specific bot

```
result: List[GameHighScore] = await bot(GetGameHighScores(...))
```

#### sendGame

Returns: Message

```
class aiogram.methods.send_game.SendGame(*, chat id: int, game short name: str,
                                                 business_connection_id: str | None = None,
                                                 message thread id: int | None = None, disable notification:
                                                 bool | None = None, protect_content: bool |
                                                 ~aiogram.client.default.Default | None =
                                                 < Default('protect content')>, message effect id: str | None =
                                                 None, reply parameters:
                                                 ~aiogram.types.reply_parameters.ReplyParameters | None =
                                                 None, reply markup:
                                                 ~aiogram.types.inline_keyboard_markup.InlineKeyboardMarkup
                                                 | None = None, allow_sending_without_reply: bool | None =
                                                 None, reply_to_message_id: int | None = None, **extra_data:
                                                 ~typing.Any)
     Use this method to send a game. On success, the sent aiogram.types.message.Message is returned.
     Source: https://core.telegram.org/bots/api#sendgame
     chat_id: int
          Unique identifier for the target chat
     game_short_name:
                           str
          Short name of the game, serves as the unique identifier for the game. Set up your games via @BotFather.
     business_connection_id: str | None
          Unique identifier of the business connection on behalf of which the message will be sent
     message_thread_id: int | None
          Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
     disable_notification: bool | None
          Sends the message silently. Users will receive a notification with no sound.
     protect_content: bool | Default | None
          Protects the contents of the sent message from forwarding and saving
     model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
          A dictionary of computed field names and their corresponding ComputedFieldInfo objects.
     model_post_init(context: Any, /) \rightarrow None
          We need to both initialize private attributes and call the user-defined model_post_init method.
     message_effect_id: str | None
          Unique identifier of the message effect to be added to the message; for private chats only
     reply_parameters: ReplyParameters | None
          Description of the message to reply to
     reply_markup: InlineKeyboardMarkup | None
          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.
     allow_sending_without_reply: bool | None
```

2.3. Bot API 527

Pass True if the message should be sent even if the specified replied-to message is not found

Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023

# reply\_to\_message\_id: int | None

If the message is a reply, ID of the original message

Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023

# **Usage**

#### As bot method

```
result: Message = await bot.send_game(...)
```

# Method as object

# Imports:

- from aiogram.methods.send\_game import SendGame
- alias: from aiogram.methods import SendGame

# With specific bot

```
result: Message = await bot(SendGame(...))
```

#### As reply into Webhook in handler

```
return SendGame(...)
```

# As shortcut from received object

- aiogram.types.message.Message.answer\_game()
- aiogram.types.message.Message.reply\_game()
- aiogram.types.chat\_join\_request.ChatJoinRequest.answer\_game()
- aiogram.types.chat\_join\_request.ChatJoinRequest.answer\_game\_pm()
- aiogram.types.chat\_member\_updated.ChatMemberUpdated.answer\_game()
- aiogram.types.inaccessible\_message.InaccessibleMessage.answer\_game()
- aiogram.types.inaccessible\_message.InaccessibleMessage.reply\_game()

#### setGameScore

Returns: Union[Message, bool]

Use this method to set the score of the specified user in a game message. On success, if the message is not an inline message, the <code>aiogram.types.message.Message</code> is returned, otherwise True is returned. Returns an error, if the new score is not greater than the user's current score in the chat and <code>force</code> is <code>False</code>.

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

```
user_id: int
```

User identifier

score: int

New score, must be non-negative

```
force: bool | None
```

Pass True if the high score is allowed to decrease. This can be useful when fixing mistakes or banning cheaters

```
disable_edit_message: bool | None
```

Pass True if the game message should not be automatically edited to include the current scoreboard

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
chat_id: int | None
```

Required if *inline\_message\_id* is not specified. Unique identifier for the target chat

```
message_id: int | None
```

Required if *inline\_message\_id* is not specified. Identifier of the sent message

```
inline_message_id: str | None
```

Required if chat\_id and message\_id are not specified. Identifier of the inline message

#### **Usage**

#### As bot method

```
result: Union[Message, bool] = await bot.set_game_score(...)
```

# Method as object

#### Imports:

- from aiogram.methods.set\_game\_score import SetGameScore
- alias: from aiogram.methods import SetGameScore

#### With specific bot

```
result: Union[Message, bool] = await bot(SetGameScore(...))
```

## As reply into Webhook in handler

```
return SetGameScore(...)
```

#### **Payments**

### answerPreCheckoutQuery

Returns: bool

class aiogram.methods.answer\_pre\_checkout\_query.AnswerPreCheckoutQuery(\*,

pre\_checkout\_query\_id:
str, ok: bool,
error\_message: str |
None = None,
\*\*extra\_data: Any)

Once the user has confirmed their payment and shipping details, the Bot API sends the final confirmation in the form of an <code>aiogram.types.update.Update</code> with the field <code>pre\_checkout\_query</code>. Use this method to respond to such pre-checkout queries. On success, <code>True</code> is returned. **Note:** The Bot API must receive an answer within 10 seconds after the pre-checkout query was sent.

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

#### pre\_checkout\_query\_id: str

Unique identifier for the query to be answered

#### ok: bool

Specify True if everything is alright (goods are available, etc.) and the bot is ready to proceed with the order. Use False if there are any problems.

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
error_message: str | None
```

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.

## Usage

#### As bot method

```
result: bool = await bot.answer_pre_checkout_query(...)
```

# Method as object

Imports:

- from aiogram.methods.answer\_pre\_checkout\_query import AnswerPreCheckoutQuery
- alias: from aiogram.methods import AnswerPreCheckoutQuery

### With specific bot

```
result: bool = await bot(AnswerPreCheckoutQuery(...))
```

# As reply into Webhook in handler

```
return AnswerPreCheckoutQuery(...)
```

### As shortcut from received object

• aiogram.types.pre\_checkout\_query.PreCheckoutQuery.answer()

# answerShippingQuery

Returns: bool

If you sent an invoice requesting a shipping address and the parameter *is\_flexible* was specified, the Bot API will send an *aiogram.types.update.Update* with a *shipping\_query* field to the bot. Use this method to reply to shipping queries. On success, True is returned.

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

## shipping\_query\_id: str

Unique identifier for the query to be answered

ok: bool

Pass 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)

### model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
shipping_options: List[ShippingOption] | None
```

Required if *ok* is True. A JSON-serialized array of available shipping options.

```
error_message: str | None
```

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.

# **Usage**

#### As bot method

```
result: bool = await bot.answer_shipping_query(...)
```

# Method as object

#### Imports:

- from aiogram.methods.answer\_shipping\_query import AnswerShippingQuery
- alias: from aiogram.methods import AnswerShippingQuery

# With specific bot

```
result: bool = await bot(AnswerShippingQuery(...))
```

# As reply into Webhook in handler

```
return AnswerShippingQuery(...)
```

# As shortcut from received object

• aiogram.types.shipping\_query.ShippingQuery.answer()

#### createInvoiceLink

Returns: str

class aiogram.methods.create\_invoice\_link.CreateInvoiceLink(\*, title: str, description: str, payload:

str, currency: str, prices: *List*[LabeledPrice], *provider\_token*:  $str \mid None = None, max\_tip\_amount$ :  $int \mid None = None$ . suggested tip amounts: List[int]  $None = None, provider\_data: str |$ None = None, photo url: str | None =*None*,  $photo\_size$ :  $int \mid None = None$ , photo width: int | None = None, photo height: int | None = None,  $need\_name: bool \mid None = None,$ need\_phone\_number: bool | None =  $None, need\_email: bool | None =$ None, need\_shipping\_address: bool | None = None, send\_phone\_number\_to\_provider:  $bool \mid None = None,$ send\_email\_to\_provider: bool | None = None, is\_flexible: bool | None = None, \*\*extra\_data: Any)

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

Source: https://core.telegram.org/bots/api#createinvoicelink

#### title: str

Product name, 1-32 characters

#### description: str

Product description, 1-255 characters

# payload: str

Bot-defined invoice payload, 1-128 bytes. This will not be displayed to the user, use it for your internal processes.

#### currency: str

Three-letter ISO 4217 currency code, see more on currencies. Pass 'XTR' for payments in Telegram Stars.

### prices: List[LabeledPrice]

Price breakdown, a JSON-serialized list of components (e.g. product price, tax, discount, delivery cost, delivery tax, bonus, etc.). Must contain exactly one item for payments in Telegram Stars.

#### provider\_token: str | None

Payment provider token, obtained via @BotFather. Pass an empty string for payments in Telegram Stars.

## max\_tip\_amount: int | None

The maximum accepted amount for tips in the *smallest units* of the currency (integer, **not** float/double). For example, for a maximum tip of US\$ 1.45 pass max\_tip\_amount = 145. See the *exp* parameter in currencies.json, it shows the number of digits past the decimal point for each currency (2 for the majority of currencies). Defaults to 0. Not supported for payments in Telegram Stars.

### suggested\_tip\_amounts: List[int] | None

A JSON-serialized array of suggested amounts of tips in the *smallest units* of the currency (integer, **not** float/double). At most 4 suggested tip amounts can be specified. The suggested tip amounts must be positive, passed in a strictly increased order and must not exceed *max\_tip\_amount*.

#### provider\_data: str | None

JSON-serialized data about the invoice, which will be shared with the payment provider. A detailed description of required fields should be provided by the payment provider.

## photo\_url: str | None

URL of the product photo for the invoice. Can be a photo of the goods or a marketing image for a service.

# model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

## $model_post_init(context: Any, /) \rightarrow None$

We need to both initialize private attributes and call the user-defined model\_post\_init method.

# photo\_size: int | None

Photo size in bytes

# photo\_width: int | None

Photo width

#### photo\_height: int | None

Photo height

# need\_name: bool | None

Pass True if you require the user's full name to complete the order. Ignored for payments in Telegram Stars.

# need\_phone\_number: bool | None

Pass True if you require the user's phone number to complete the order. Ignored for payments in Telegram Stars.

#### need\_email: bool | None

Pass True if you require the user's email address to complete the order. Ignored for payments in Telegram Stars.

# need\_shipping\_address: bool | None

Pass True if you require the user's shipping address to complete the order. Ignored for payments in Telegram Stars.

#### send\_phone\_number\_to\_provider: bool | None

Pass True if the user's phone number should be sent to the provider. Ignored for payments in Telegram Stars.

## send\_email\_to\_provider: bool | None

Pass True if the user's email address should be sent to the provider. Ignored for payments in Telegram Stars.

#### is\_flexible: bool | None

Pass True if the final price depends on the shipping method. Ignored for payments in Telegram Stars.

#### **Usage**

#### As bot method

```
result: str = await bot.create_invoice_link(...)
```

## Method as object

Imports:

- from aiogram.methods.create\_invoice\_link import CreateInvoiceLink
- alias: from aiogram.methods import CreateInvoiceLink

#### With specific bot

```
result: str = await bot(CreateInvoiceLink(...))
```

# As reply into Webhook in handler

```
return CreateInvoiceLink(...)
```

## getStarTransactions

Returns: StarTransactions

Returns the bot's Telegram Star transactions in chronological order. On success, returns a aiogram.types.star\_transactions.StarTransactions object.

Source: https://core.telegram.org/bots/api#getstartransactions

```
offset: int | None
```

Number of transactions to skip in the response

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
limit: int | None
```

The maximum number of transactions to be retrieved. Values between 1-100 are accepted. Defaults to 100.

#### **Usage**

#### As bot method

```
result: StarTransactions = await bot.get_star_transactions(...)
```

## Method as object

Imports:

- from aiogram.methods.get\_star\_transactions import GetStarTransactions
- alias: from aiogram.methods import GetStarTransactions

#### With specific bot

```
result: StarTransactions = await bot(GetStarTransactions(...))
```

# refundStarPayment

```
Returns: bool
```

Refunds a successful payment in Telegram Stars. Returns True on success.

Source: https://core.telegram.org/bots/api#refundstarpayment

```
user_id: int
```

Identifier of the user whose payment will be refunded

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
telegram_payment_charge_id: str
```

Telegram payment identifier

# **Usage**

#### As bot method

```
result: bool = await bot.refund_star_payment(...)
```

## Method as object

#### Imports:

- from aiogram.methods.refund\_star\_payment import RefundStarPayment
- alias: from aiogram.methods import RefundStarPayment

#### With specific bot

```
result: bool = await bot(RefundStarPayment(...))
```

## As reply into Webhook in handler

```
return RefundStarPayment(...)
```

#### sendInvoice

Returns: Message

**class** aiogram.methods.send\_invoice.**SendInvoice**(\*, chat\_id: int | str, title: str, description: str, payload: str, currency: str, prices: ~typ-

> ing.List[~aiogram.types.labeled\_price.LabeledPrice], *message\_thread\_id: int | None = None, provider\_token:* str | None = None, max\_tip\_amount: int | None = None, suggested tip amounts: ~typing.List[int] | None = None,  $start\_parameter: str | None = None,$ provider data:  $str \mid None = None$ , photo  $url: str \mid None$ = None, photo\_size: int | None = None, photo\_width: *int* | *None* = *None*, *photo height: int* | *None* = *None*, need\_name: bool | None = None, need\_phone\_number:  $bool \mid None = None, need\_email: bool \mid None = None,$  $need\_shipping\_address: bool \mid None = None,$

> *send\_phone\_number\_to\_provider: bool* | *None* = *None*,  $send\_email\_to\_provider: bool \mid None = None,$

*is\_flexible: bool* | *None* = *None, disable\_notification:* 

bool | None = None, protect\_content: bool |

~aiogram.client.default.Default | None = <Default('protect\_content')>, message\_effect\_id: str |

 $None = None, reply\_parameters:$ 

~aiogram.types.reply\_parameters.ReplyParameters |

 $None = None, reply\_markup$ :

~aiogram.types.inline\_keyboard\_markup.InlineKeyboardMarkup | None = None, allow\_sending\_without\_reply: bool |

 $None = None, reply\_to\_message\_id: int | None = None,$ \*\*extra\_data: ~typing.Any)

Use this method to send invoices. On success, the sent aiogram.types.message.Message is returned.

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

#### chat\_id: int | str

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

#### title: str

Product name, 1-32 characters

#### description: str

Product description, 1-255 characters

#### payload: str

Bot-defined invoice payload, 1-128 bytes. This will not be displayed to the user, use it for your internal processes.

#### currency: str

Three-letter ISO 4217 currency code, see more on currencies. Pass 'XTR' for payments in Telegram Stars.

#### prices: List[LabeledPrice]

Price breakdown, a JSON-serialized list of components (e.g. product price, tax, discount, delivery cost, delivery tax, bonus, etc.). Must contain exactly one item for payments in Telegram Stars.

#### message\_thread\_id: int | None

Unique identifier for the target message thread (topic) of the forum; for forum supergroups only

### provider\_token: str | None

Payment provider token, obtained via @BotFather. Pass an empty string for payments in Telegram Stars.

#### max\_tip\_amount: int | None

The maximum accepted amount for tips in the *smallest units* of the currency (integer, **not** float/double). For example, for a maximum tip of US\$ 1.45 pass max\_tip\_amount = 145. See the *exp* parameter in currencies.json, it shows the number of digits past the decimal point for each currency (2 for the majority of currencies). Defaults to 0. Not supported for payments in Telegram Stars.

## suggested\_tip\_amounts: List[int] | None

A JSON-serialized array of suggested amounts of tips in the *smallest units* of the currency (integer, **not** float/double). At most 4 suggested tip amounts can be specified. The suggested tip amounts must be positive, passed in a strictly increased order and must not exceed *max\_tip\_amount*.

#### start\_parameter: str | None

Unique deep-linking parameter. If left empty, **forwarded copies** of the sent message will have a *Pay* button, allowing multiple users to pay directly from the forwarded message, using the same invoice. If non-empty, forwarded copies of the sent message will have a *URL* button with a deep link to the bot (instead of a *Pay* button), with the value used as the start parameter

#### provider\_data: str | None

JSON-serialized data about the invoice, which will be shared with the payment provider. A detailed description of required fields should be provided by the payment provider.

#### photo\_url: str | None

URL of the product photo for the invoice. Can be a photo of the goods or a marketing image for a service. People like it better when they see what they are paying for.

#### photo\_size: int | None

Photo size in bytes

#### photo\_width: int | None

Photo width

## model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

### $model_post_init(context: Any, /) \rightarrow None$

We need to both initialize private attributes and call the user-defined model\_post\_init method.

## photo\_height: int | None

Photo height

#### need\_name: bool | None

Pass True if you require the user's full name to complete the order. Ignored for payments in Telegram Stars.

## need\_phone\_number: bool | None

Pass True if you require the user's phone number to complete the order. Ignored for payments in Telegram Stars.

#### need\_email: bool | None

Pass True if you require the user's email address to complete the order. Ignored for payments in Telegram Stars.

#### need\_shipping\_address: bool | None

Pass True if you require the user's shipping address to complete the order. Ignored for payments in Telegram Stars.

## send\_phone\_number\_to\_provider: bool | None

Pass True if the user's phone number should be sent to the provider. Ignored for payments in Telegram Stars.

#### send\_email\_to\_provider: bool | None

Pass True if the user's email address should be sent to the provider. Ignored for payments in Telegram Stars.

#### is\_flexible: bool | None

Pass True if the final price depends on the shipping method. Ignored for payments in Telegram Stars.

#### disable\_notification: bool | None

Sends the message silently. Users will receive a notification with no sound.

#### protect\_content: bool | Default | None

Protects the contents of the sent message from forwarding and saving

#### message\_effect\_id: str | None

Unique identifier of the message effect to be added to the message; for private chats only

#### reply\_parameters: ReplyParameters | None

Description of the message to reply to

#### reply\_markup: InlineKeyboardMarkup | None

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.

#### allow\_sending\_without\_reply: bool | None

Pass True if the message should be sent even if the specified replied-to message is not found

Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023

#### reply\_to\_message\_id: int | None

If the message is a reply, ID of the original message

Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023

#### Usage

#### As bot method

```
result: Message = await bot.send_invoice(...)
```

# Method as object

## Imports:

- from aiogram.methods.send\_invoice import SendInvoice
- alias: from aiogram.methods import SendInvoice

#### With specific bot

```
result: Message = await bot(SendInvoice(...))
```

# As reply into Webhook in handler

```
return SendInvoice(...)
```

#### As shortcut from received object

- aiogram.types.message.Message.answer\_invoice()
- aiogram.types.message.Message.reply\_invoice()
- aiogram.types.chat\_join\_request.ChatJoinRequest.answer\_invoice()
- aiogram.types.chat\_join\_request.ChatJoinRequest.answer\_invoice\_pm()
- aiogram.types.chat\_member\_updated.ChatMemberUpdated.answer\_invoice()
- aiogram.types.inaccessible\_message.InaccessibleMessage.answer\_invoice()
- aiogram.types.inaccessible\_message.InaccessibleMessage.reply\_invoice()

## **Getting updates**

## deleteWebhook

Returns: bool

Use this method to remove webhook integration if you decide to switch back to aiogram.methods. get\_updates.GetUpdates. Returns True on success.

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

```
drop_pending_updates: bool | None
```

Pass True to drop all pending updates

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

## **Usage**

#### As bot method

```
result: bool = await bot.delete_webhook(...)
```

## Method as object

#### Imports:

- from aiogram.methods.delete\_webhook import DeleteWebhook
- alias: from aiogram.methods import DeleteWebhook

# With specific bot

```
result: bool = await bot(DeleteWebhook(...))
```

# As reply into Webhook in handler

```
return DeleteWebhook(...)
```

## getUpdates

```
Returns: List[Update]
```

```
class aiogram.methods.get_updates.GetUpdates(*, offset: int | None = None, limit: int | None = None, timeout: int | None = None, allowed_updates: List[str] | None = None, **extra_data: Any)
```

Use this method to receive incoming updates using long polling (wiki). Returns an Array of aiogram.types.update.Update objects.

#### 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

#### offset: int | None

Identifier of the first update to be returned. Must be greater by one than the highest among the identifiers of previously received updates. By default, updates starting with the earliest unconfirmed update are returned. An update is considered confirmed as soon as <code>aiogram.methods.get\_updates.GetUpdates</code> is called with an <code>offset</code> higher than its <code>update\_id</code>. The negative offset can be specified to retrieve updates starting from <code>-offset</code> update from the end of the updates queue. All previous updates will be forgotten.

#### limit: int | None

Limits the number of updates to be retrieved. Values between 1-100 are accepted. Defaults to 100.

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### timeout: int | None

Timeout in seconds for long polling. Defaults to 0, i.e. usual short polling. Should be positive, short polling should be used for testing purposes only.

# allowed\_updates: List[str] | None

A JSON-serialized list of the update types you want your bot to receive. For example, specify ["message", "edited\_channel\_post", "callback\_query"] to only receive updates of these types. See aiogram. types.update.Update for a complete list of available update types. Specify an empty list to receive all update types except chat\_member, message\_reaction, and message\_reaction\_count (default). If not specified, the previous setting will be used.

#### Usage

#### As bot method

```
result: List[Update] = await bot.get_updates(...)
```

## Method as object

#### Imports:

- from aiogram.methods.get\_updates import GetUpdates
- alias: from aiogram.methods import GetUpdates

#### With specific bot

```
result: List[Update] = await bot(GetUpdates(...))
```

#### getWebhookInfo

Returns: WebhookInfo

class aiogram.methods.get\_webhook\_info.GetWebhookInfo(\*\*extra\_data: Any)

Use this method to get current webhook status. Requires no parameters. On success, returns a <code>aiogram.types.webhook\_info.WebhookInfo</code> object. If the bot is using <code>aiogram.methods.get\_updates.GetUpdates</code>, will return an object with the <code>url</code> field empty.

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

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### **Usage**

#### As bot method

```
result: WebhookInfo = await bot.get_webhook_info(...)
```

## Method as object

Imports:

- from aiogram.methods.get\_webhook\_info import GetWebhookInfo
- alias: from aiogram.methods import GetWebhookInfo

#### With specific bot

```
result: WebhookInfo = await bot(GetWebhookInfo(...))
```

#### setWebhook

Returns: bool

```
 \textbf{class} \  \, \textbf{aiogram.methods.set\_webhook}(*, \textit{url: str, certificate: } \  \, \textbf{InputFile} \mid \textit{None} = \textit{None}, \\ ip\_address: \textit{str} \mid \textit{None} = \textit{None, max\_connections: int} \mid \\ None = \textit{None, allowed\_updates: } \  \, \textit{List[str]} \mid \textit{None} = \textit{None,} \\ drop\_pending\_updates: \textit{bool} \mid \textit{None} = \textit{None,} \\ secret\_token: \textit{str} \mid \textit{None} = \textit{None, **extra\_data: Any})
```

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 <code>aiogram.types.update.Update</code>. In case of an unsuccessful request, we will give up after a reasonable amount of attempts. Returns True on success. If you'd like to make sure that the webhook was set by you, you can specify secret data in the parameter <code>secret\_token</code>. If specified, the request will contain a header 'X-Telegram-Bot-Api-Secret-Token' with the secret token as content.

#### **Notes**

- **1.** You will not be able to receive updates using aiogram.methods.get\_updates.GetUpdates for as long as an outgoing webhook is set up.
- **2.** To use a self-signed certificate, you need to upload your public key certificate using *certificate* parameter. Please upload as InputFile, sending a String will not work.
- **3.** Ports currently supported *for webhooks*: **443**, **80**, **88**, **8443**. If you're having any trouble setting up webhooks, please check out this amazing guide to webhooks.

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

#### url: str

HTTPS URL to send updates to. Use an empty string to remove webhook integration

## certificate: InputFile | None

Upload your public key certificate so that the root certificate in use can be checked. See our self-signed guide for details.

#### ip\_address: str | None

The fixed IP address which will be used to send webhook requests instead of the IP address resolved through DNS

#### max\_connections: int | None

The maximum allowed number of simultaneous HTTPS connections to the webhook for update delivery, 1-100. Defaults to 40. Use lower values to limit the load on your bot's server, and higher values to increase your bot's throughput.

## model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

#### $model_post_init(context: Any, /) \rightarrow None$

We need to both initialize private attributes and call the user-defined model\_post\_init method.

#### allowed\_updates: List[str] | None

A JSON-serialized list of the update types you want your bot to receive. For example, specify ["message", "edited\_channel\_post", "callback\_query"] to only receive updates of these types. See aiogram. types.update.Update for a complete list of available update types. Specify an empty list to receive all update types except chat\_member, message\_reaction, and message\_reaction\_count (default). If not specified, the previous setting will be used.

#### drop\_pending\_updates: bool | None

Pass True to drop all pending updates

## secret\_token: str | None

A secret token to be sent in a header 'X-Telegram-Bot-Api-Secret-Token' in every webhook request, 1-256 characters. Only characters A-Z, a-z, 0-9, \_ and - are allowed. The header is useful to ensure that the request comes from a webhook set by you.

#### Usage

#### As bot method

```
result: bool = await bot.set_webhook(...)
```

# Method as object

Imports:

- from aiogram.methods.set\_webhook import SetWebhook
- alias: from aiogram.methods import SetWebhook

#### With specific bot

```
result: bool = await bot(SetWebhook(...))
```

# As reply into Webhook in handler

```
return SetWebhook(...)
```

# **Telegram Passport**

# setPassportDataErrors

Returns: bool

class aiogram.methods.set\_passport\_data\_errors.SetPassportDataErrors(\*, user\_id: int, errors:

List/PassportElementErrorDataField | PassportElementError-FrontSide | PassportElementErrorReverseSide | PassportElementError-Selfie | PassportElementErrorFile | PassportElementError-Files PassportElementError-TranslationFile | PassportElementError-TranslationFiles | PassportElementErrorUnspecified], \*\*extra\_data:

Any)

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

#### user id: int

User identifier

## model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
errors: List[PassportElementErrorDataField | PassportElementErrorFrontSide |
PassportElementErrorReverseSide | PassportElementErrorSelfie |
PassportElementErrorFile | PassportElementErrorFiles |
PassportElementErrorTranslationFile | PassportElementErrorTranslationFiles |
PassportElementErrorUnspecified]
```

A JSON-serialized array describing the errors

## Usage

#### As bot method

```
result: bool = await bot.set_passport_data_errors(...)
```

## Method as object

Imports:

- from aiogram.methods.set\_passport\_data\_errors import SetPassportDataErrors
- alias: from aiogram.methods import SetPassportDataErrors

#### With specific bot

```
result: bool = await bot(SetPassportDataErrors(...))
```

#### As reply into Webhook in handler

```
return SetPassportDataErrors(...)
```

# 2.3.5 **Enums**

Here is list of all available enums:

## **BotCommandScopeType**

This object represents the scope to which bot commands are applied.

Source: https://core.telegram.org/bots/api#botcommandscope

```
DEFAULT = 'default'
ALL_PRIVATE_CHATS = 'all_private_chats'
ALL_GROUP_CHATS = 'all_group_chats'
ALL_CHAT_ADMINISTRATORS = 'all_chat_administrators'
CHAT = 'chat'
CHAT_ADMINISTRATORS = 'chat_administrators'
CHAT_MEMBER = 'chat_member'
```

#### **ChatAction**

This object represents bot actions.

Choose one, depending on what the user is about to receive:

- · typing for text messages,
- upload\_photo for photos,
- record\_video or upload\_video for videos,
- record\_voice or upload\_voice for voice notes,
- upload\_document for general files,
- · choose\_sticker for stickers,
- find\_location for location data,
- record\_video\_note or upload\_video\_note for video notes.

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

```
TYPING = 'typing'
UPLOAD_PHOTO = 'upload_photo'
RECORD_VIDEO = 'record_video'
```

```
UPLOAD_VIDEO = 'upload_video'
RECORD_VOICE = 'record_voice'
UPLOAD_VOICE = 'upload_voice'
UPLOAD_DOCUMENT = 'upload_document'
CHOOSE_STICKER = 'choose_sticker'
FIND_LOCATION = 'find_location'
RECORD_VIDEO_NOTE = 'record_video_note'
UPLOAD_VIDEO_NOTE = 'upload_video_note'
```

## ChatBoostSourceType

This object represents a type of chat boost source.

Source: https://core.telegram.org/bots/api#chatboostsource

```
PREMIUM = 'premium'

GIFT_CODE = 'gift_code'

GIVEAWAY = 'giveaway'
```

#### ChatMemberStatus

```
 \textbf{class} \  \, \textbf{aiogram.enums.chat\_member\_status.ChatMemberStatus}(\textit{value}, \textit{names=None}, *, \textit{module=None}, \\ \textit{qualname=None}, \textit{type=None}, \textit{start=1}, \\ \textit{boundary=None})
```

This object represents chat member status.

Source: https://core.telegram.org/bots/api#chatmember

```
CREATOR = 'creator'
ADMINISTRATOR = 'administrator'
MEMBER = 'member'
RESTRICTED = 'restricted'
LEFT = 'left'
KICKED = 'kicked'
```

# ChatType

# ContentType

**class** aiogram.enums.content\_type.**ContentType**(*value*, *names=None*, \*, *module=None*, *qualname=None*, type=None, start=1, boundary=None)

This object represents a type of content in message

```
UNKNOWN = 'unknown'
ANY = 'any'
TEXT = 'text'
ANIMATION = 'animation'
AUDIO = 'audio'
DOCUMENT = 'document'
PAID_MEDIA = 'paid_media'
PHOTO = 'photo'
STICKER = 'sticker'
STORY = 'story'
VIDEO = 'video'
VIDEO_NOTE = 'video_note'
VOICE = 'voice'
CONTACT = 'contact'
DICE = 'dice'
GAME = 'game'
POLL = 'poll'
```

```
VENUE = 'venue'
LOCATION = 'location'
NEW_CHAT_MEMBERS = 'new_chat_members'
LEFT_CHAT_MEMBER = 'left_chat_member'
NEW_CHAT_TITLE = 'new_chat_title'
NEW_CHAT_PHOTO = 'new_chat_photo'
DELETE_CHAT_PHOTO = 'delete_chat_photo'
GROUP_CHAT_CREATED = 'group_chat_created'
SUPERGROUP_CHAT_CREATED = 'supergroup_chat_created'
CHANNEL_CHAT_CREATED = 'channel_chat_created'
MESSAGE_AUTO_DELETE_TIMER_CHANGED = 'message_auto_delete_timer_changed'
MIGRATE_TO_CHAT_ID = 'migrate_to_chat_id'
MIGRATE_FROM_CHAT_ID = 'migrate_from_chat_id'
PINNED_MESSAGE = 'pinned_message'
INVOICE = 'invoice'
SUCCESSFUL_PAYMENT = 'successful_payment'
REFUNDED_PAYMENT = 'refunded_payment'
USERS_SHARED = 'users_shared'
CHAT_SHARED = 'chat_shared'
CONNECTED_WEBSITE = 'connected_website'
WRITE_ACCESS_ALLOWED = 'write_access_allowed'
PASSPORT_DATA = 'passport_data'
PROXIMITY_ALERT_TRIGGERED = 'proximity_alert_triggered'
BOOST_ADDED = 'boost_added'
CHAT_BACKGROUND_SET = 'chat_background_set'
FORUM_TOPIC_CREATED = 'forum_topic_created'
FORUM_TOPIC_EDITED = 'forum_topic_edited'
FORUM_TOPIC_CLOSED = 'forum_topic_closed'
FORUM_TOPIC_REOPENED = 'forum_topic_reopened'
GENERAL_FORUM_TOPIC_HIDDEN = 'general_forum_topic_hidden'
GENERAL_FORUM_TOPIC_UNHIDDEN = 'general_forum_topic_unhidden'
```

```
GIVEAWAY_CREATED = 'giveaway_created'

GIVEAWAY = 'giveaway'

GIVEAWAY_WINNERS = 'giveaway_winners'

GIVEAWAY_COMPLETED = 'giveaway_completed'

VIDEO_CHAT_SCHEDULED = 'video_chat_scheduled'

VIDEO_CHAT_STARTED = 'video_chat_started'

VIDEO_CHAT_ENDED = 'video_chat_ended'

VIDEO_CHAT_PARTICIPANTS_INVITED = 'video_chat_participants_invited'

WEB_APP_DATA = 'web_app_data'

USER_SHARED = 'user_shared'
```

## Currency

**class** aiogram.enums.currency.**Currency**(*value*, *names=None*, \*, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Currencies supported by Telegram Bot API

Source: https://core.telegram.org/bots/payments#supported-currencies

AED = 'AED'

AFN = 'AFN'

ALL = 'ALL'

AMD = 'AMD'

ARS = 'ARS'

AUD = 'AUD'

AZN = 'AZN'

BAM = 'BAM'

BDT = 'BDT'

BGN = 'BGN'

BND = 'BND'

BOB = 'BOB'

BRL = 'BRL'

BYN = 'BYN'

CAD = 'CAD'

CHF = 'CHF'

- CLP = 'CLP'
- CNY = 'CNY'
- COP = 'COP'
- CRC = 'CRC'
- CZK = 'CZK'
- DKK = 'DKK'
- DOP = 'DOP'
- DZD = 'DZD'
- EGP = 'EGP'
- ETB = 'ETB'
- EUR = 'EUR'
- GBP = 'GBP'
- GEL = 'GEL'
- GTQ = 'GTQ'
- HKD = 'HKD'
- HNL = 'HNL'
- HRK = 'HRK'
- HUF = 'HUF'
- IDR = 'IDR'
- ILS = 'ILS'
- INR = 'INR'
- ISK = 'ISK'
- JMD = 'JMD'
- JPY = 'JPY'
- KES = 'KES'
- KGS = 'KGS'
- KRW = 'KRW'
- KZT = 'KZT'
- LBP = 'LBP'
- LKR = 'LKR'
- MAD = 'MAD'

MDL = 'MDL'

MNT = 'MNT'

MUR = 'MUR'

MVR = 'MVR'

MXN = 'MXN'

MYR = 'MYR'

MZN = 'MZN'

NGN = 'NGN'

NIO = 'NIO'

NOK = 'NOK'

NPR = 'NPR'

NZD = 'NZD'

PAB = 'PAB'

PEN = 'PEN'

PHP = 'PHP'

PKR = 'PKR'

PLN = 'PLN'

PYG = 'PYG'

QAR = 'QAR'

RON = 'RON'

RSD = 'RSD'

RUB = 'RUB'

SAR = 'SAR'

SEK = 'SEK'

SGD = 'SGD'

THB = 'THB'

TJS = 'TJS'

TRY = 'TRY'

TTD = 'TTD'

TWD = 'TWD'

TZS = 'TZS'

```
UAH = 'UAH'

UGX = 'UGX'

USD = 'USD'

UYU = 'UYU'

UZS = 'UZS'

VND = 'VND'

YER = 'YER'

ZAR = 'ZAR'
```

# **DiceEmoji**

Emoji on which the dice throw animation is based

```
Source: https://core.telegram.org/bots/api#dice
```

```
DICE = ''

DART = ''

BASKETBALL = ''

FOOTBALL = ''

SLOT_MACHINE = ''

BOWLING = ''
```

## **EncryptedPassportElement**

This object represents type of encrypted passport element.

Source: https://core.telegram.org/bots/api#encrypted pass portelement

```
PERSONAL_DETAILS = 'personal_details'

PASSPORT = 'passport'

DRIVER_LICENSE = 'driver_license'

IDENTITY_CARD = 'identity_card'

INTERNAL_PASSPORT = 'internal_passport'
```

```
UTILITY_BILL = 'utility_bill'
     BANK_STATEMENT = 'bank_statement'
     RENTAL_AGREEMENT = 'rental_agreement'
     PASSPORT_REGISTRATION = 'passport_registration'
     TEMPORARY_REGISTRATION = 'temporary_registration'
     PHONE_NUMBER = 'phone_number'
     EMAIL = 'email'
InlineQueryResultType
class aiogram.enums.inline_query_result_type.InlineQueryResultType(value, names=None, *,
                                                                       module=None,
                                                                       qualname=None,
                                                                       type=None, start=1,
                                                                       boundary=None)
     Type of inline query result
     Source: https://core.telegram.org/bots/api#inlinequeryresult
     AUDIO = 'audio'
     DOCUMENT = 'document'
     GIF = 'gif'
     MPEG4_GIF = 'mpeg4_gif'
     PHOTO = 'photo'
     STICKER = 'sticker'
     VIDEO = 'video'
     VOICE = 'voice'
     ARTICLE = 'article'
     CONTACT = 'contact'
     GAME = 'game'
     LOCATION = 'location'
     VENUE = 'venue'
```

ADDRESS = 'address'

## InputMediaType

## InputPaidMediaType

This object represents the type of a media in a paid message.

```
Source: https://core.telegram.org/bots/api#inputpaidmedia
```

```
PHOTO = 'photo'
VIDEO = 'video'
```

#### KeyboardButtonPollTypeType

 ${\bf class} \ {\bf aiogram.enums.keyboard\_button\_poll\_type\_type.KeyboardButtonPollTypeType(\it value, \it va$ 

```
names=None,
*,
module=None,
qual-
name=None,
type=None,
start=1,
bound-
ary=None)
```

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

```
Source: https://core.telegram.org/bots/api#keyboardbuttonpolltype
```

```
QUIZ = 'quiz'

REGULAR = 'regular'
```

## **MaskPositionPoint**

The part of the face relative to which the mask should be placed.

Source: https://core.telegram.org/bots/api#maskposition

```
FOREHEAD = 'forehead'

EYES = 'eyes'

MOUTH = 'mouth'

CHIN = 'chin'
```

# MenuButtonType

This object represents an type of Menu button

Source: https://core.telegram.org/bots/api#menubuttondefault

```
DEFAULT = 'default'
COMMANDS = 'commands'
WEB_APP = 'web_app'
```

#### MessageEntityType

This object represents type of message entity

Source: https://core.telegram.org/bots/api#messageentity

```
MENTION = 'mention'

HASHTAG = 'hashtag'

CASHTAG = 'cashtag'

BOT_COMMAND = 'bot_command'

URL = 'url'

EMAIL = 'email'

PHONE_NUMBER = 'phone_number'

BOLD = 'bold'
```

```
ITALIC = 'italic'
UNDERLINE = 'underline'
STRIKETHROUGH = 'strikethrough'
SPOILER = 'spoiler'
BLOCKQUOTE = 'blockquote'
EXPANDABLE_BLOCKQUOTE = 'expandable_blockquote'
CODE = 'code'
PRE = 'pre'
TEXT_LINK = 'text_link'
TEXT_MENTION = 'text_mention'
CUSTOM_EMOJI = 'custom_emoji'
```

# MessageOriginType

```
\begin{tabular}{ll} \textbf{class} & aiogram.enums.message\_origin\_type. \textbf{MessageOriginType}(value, names=None, *, module=None, qualname=None, type=None, start=1, boundary=None) \end{tabular}
```

This object represents origin of a message.

```
Source: https://core.telegram.org/bots/api#messageorigin
```

```
USER = 'user'
HIDDEN_USER = 'hidden_user'
CHAT = 'chat'
CHANNEL = 'channel'
```

#### **PaidMediaType**

This object represents the type of a media in a paid message.

Source: https://core.telegram.org/bots/api#paidmedia

```
PHOTO = 'photo'

PREVIEW = 'preview'

VIDEO = 'video'
```

## **ParseMode**

#### **PassportElementErrorType**

This object represents a passport element error type.

```
\textbf{Source:}\ https://core.telegram.org/bots/api\#passportelementerror
```

```
DATA = 'data'

FRONT_SIDE = 'front_side'

REVERSE_SIDE = 'reverse_side'

SELFIE = 'selfie'

FILE = 'file'

FILES = 'files'

TRANSLATION_FILE = 'translation_file'

TRANSLATION_FILES = 'translation_files'

UNSPECIFIED = 'unspecified'
```

## **PoliType**

QUIZ = 'quiz'

## ReactionTypeType

```
\begin{tabular}{ll} \textbf{class} & aiogram.enums.reaction\_type\_type. \textbf{ReactionTypeType}(value, names=None, *, module=None, qualname=None, type=None, start=1, boundary=None) \end{tabular}
```

This object represents reaction type.

```
Source: https://core.telegram.org/bots/api#reactiontype
```

```
EMOJI = 'emoji'
CUSTOM_EMOJI = 'custom_emoji'
```

## RevenueWithdrawalStateType

```
class aiogram.enums.revenue_withdrawal_state_type.RevenueWithdrawalStateType(value,
```

names=None,
\*,
module=None,
qualname=None,
type=None,
start=1, boundary=None)

This object represents a revenue withdrawal state type

Source: https://core.telegram.org/bots/api#revenuewithdrawalstate

```
FAILED = 'failed'
PENDING = 'pending'
SUCCEEDED = 'succeeded'
```

#### **StickerFormat**

Format of the sticker

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

```
STATIC = 'static'

ANIMATED = 'animated'

VIDEO = 'video'
```

## **StickerType**

The part of the face relative to which the mask should be placed.

Source: https://core.telegram.org/bots/api#maskposition

```
REGULAR = 'regular'
MASK = 'mask'
CUSTOM_EMOJI = 'custom_emoji'
```

#### **TopicIconColor**

Color of the topic icon in RGB format.

Source: https://github.com/telegramdesktop/tdesktop/blob/991fe491c5ae62705d77aa8fdd44a79caf639c45/Telegram/SourceFiles/data/data\_forum\_topic.cpp#L51-L56

```
BLUE = 7322096

YELLOW = 16766590

VIOLET = 13338331

GREEN = 9367192

ROSE = 16749490

RED = 16478047
```

# **TransactionPartnerType**

This object represents a type of transaction partner.

Source: https://core.telegram.org/bots/api#transactionpartner

```
FRAGMENT = 'fragment'
OTHER = 'other'
USER = 'user'
TELEGRAM_ADS = 'telegram_ads'
```

## **UpdateType**

```
type=None, start=1, boundary=None)
This object represents the complete list of allowed update types
Source: https://core.telegram.org/bots/api#update
MESSAGE = 'message'
EDITED_MESSAGE = 'edited_message'
CHANNEL_POST = 'channel_post'
EDITED_CHANNEL_POST = 'edited_channel_post'
BUSINESS_CONNECTION = 'business_connection'
BUSINESS_MESSAGE = 'business_message'
EDITED_BUSINESS_MESSAGE = 'edited_business_message'
DELETED_BUSINESS_MESSAGES = 'deleted_business_messages'
MESSAGE_REACTION = 'message_reaction'
MESSAGE_REACTION_COUNT = 'message_reaction_count'
INLINE_QUERY = 'inline_query'
CHOSEN_INLINE_RESULT = 'chosen_inline_result'
CALLBACK_QUERY = 'callback_query'
SHIPPING_QUERY = 'shipping_query'
PRE_CHECKOUT_QUERY = 'pre_checkout_query'
PURCHASED_PAID_MEDIA = 'purchased_paid_media'
POLL = 'poll'
POLL_ANSWER = 'poll_answer'
MY_CHAT_MEMBER = 'my_chat_member'
CHAT_MEMBER = 'chat_member'
CHAT_JOIN_REQUEST = 'chat_join_request'
CHAT_BOOST = 'chat_boost'
REMOVED_CHAT_BOOST = 'removed_chat_boost'
```

class aiogram.enums.update\_type.UpdateType(value, names=None, \*, module=None, qualname=None,

# 2.3.6 How to download file?

## **Download file manually**

First, you must get the *file\_id* of the file you want to download. Information about files sent to the bot is contained in Message.

For example, download the document that came to the bot.

```
file_id = message.document.file_id
```

Then use the getFile method to get *file path*.

```
file = await bot.get_file(file_id)
file_path = file.file_path
```

After that, use the *download\_file* method from the bot object.

#### download\_file(...)

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.

**async** Bot.download\_file(file\_path: str, destination: BinaryIO | Path | str | None = None, timeout: int = 30, chunk\_size: int = 65536, seek: bool = True)  $\rightarrow$  BinaryIO | None

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 File path on Telegram server (You can get it from aiogram.types.File)
- **destination** Filename, file path or instance of io.IOBase. For e.g. io.BytesIO, defaults to None
- timeout Total timeout in seconds, defaults to 30
- **chunk\_size** File chunks size, defaults to 64 kb
- **seek** Go to start of file when downloading is finished. Used only for destination with typing.BinaryIO type, defaults to True

There are two options where you can download the file: to disk or to binary I/O object.

#### Download file to disk

To download file to disk, you must specify the file name or path where to download the file. In this case, the function will return nothing.

```
await bot.download_file(file_path, "text.txt")
```

#### Download file to binary I/O object

To download file to binary I/O object, you must specify an object with the typing.BinaryIO type or use the default (None) value.

In the first case, the function will return your object:

```
my_object = MyBinaryIO()
result: MyBinaryIO = await bot.download_file(file_path, my_object)
# print(result is my_object) # True
```

If you leave the default value, an io.BytesIO object will be created and returned.

```
result: io.BytesIO = await bot.download_file(file_path)
```

# Download file in short way

Getting file\_path manually every time is boring, so you should use the download method.

#### download(...)

Download file by file\_id or Downloadable object to destination.

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

**async** Bot.download(file:  $str \mid Downloadable$ , destination:  $BinaryIO \mid Path \mid str \mid None = None$ , timeout: int = 30,  $chunk \ size$ : int = 65536, seek: bool = True)  $\rightarrow$  BinaryIO | None

Download file by file\_id or Downloadable object to destination.

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

#### **Parameters**

- **file** file\_id or Downloadable object
- **destination** Filename, file path or instance of io.IOBase. For e.g. io.BytesIO, defaults to None
- timeout Total timeout in seconds, defaults to 30
- **chunk\_size** File chunks size, defaults to 64 kb
- **seek** Go to start of file when downloading is finished. Used only for destination with typing.BinaryIO type, defaults to True

It differs from *download\_file* **only** in that it accepts *file\_id* or an *Downloadable* object (object that contains the *file\_id* attribute) instead of *file\_path*.

You can download a file to *disk* or to a *binary I/O* object in the same way.

Example:

```
document = message.document
await bot.download(document)
```

# 2.3.7 How to upload file?

As says official Telegram Bot API documentation there are three ways to send files (photos, stickers, audio, media, etc.):

If the file is already stored somewhere on the Telegram servers or file is available by the URL, you don't need to reupload it.

But if you need to upload a new file just use subclasses of InputFile.

Here are the three different available builtin types of input file:

- aiogram.types.input\_file.FSInputFile uploading from file system
- aiogram.types.input\_file.BufferedInputFile uploading from buffer
- aiogram.types.input\_file.URLInputFile uploading from URL

# **A** Warning

#### Be respectful to Telegram

Instances of *InputFile* are reusable. That means you can create an instance of InputFile and send it multiple times. However, Telegram does not recommend doing this. Instead, once you upload a file, save its *file\_id* and reuse that later.

## Upload from file system

By first step you will need to import InputFile wrapper:

```
from aiogram.types import FSInputFile
```

Then you can use it:

```
cat = FSInputFile("cat.png")
agenda = FSInputFile("my-document.pdf", filename="agenda-2019-11-19.pdf")
```

\_\_init\_\_(path: str | Path, filename: str | None = None, chunk\_size: int = 65536)

Represents object for uploading files from filesystem

#### **Parameters**

• path – Path to file

- **filename** Filename to be propagated to telegram. By default, will be parsed from path
- **chunk\_size** Uploading chunk size

#### **Upload from buffer**

Files can be also passed from buffer (For example you generate image using Pillow and you want to send it to Telegram): Import wrapper:

```
from aiogram.types import BufferedInputFile
```

And then you can use it:

```
text_file = BufferedInputFile(b"Hello, world!", filename="file.txt")
```

class aiogram.types.input\_file.BufferedInputFile(file: bytes, filename: str, chunk\_size: int = 65536)

```
__init__(file: bytes, filename: str, chunk_size: int = 65536)
```

Represents object for uploading files from filesystem

#### **Parameters**

- file Bytes
- **filename** Filename to be propagated to telegram.
- chunk\_size Uploading chunk size

# **Upload from url**

If you need to upload a file from another server, but the direct link is bound to your server's IP, or you want to bypass native upload limits by URL, you can use <code>aiogram.types.input\_file.URLInputFile</code>.

Import wrapper:

```
from aiogram.types import URLInputFile
```

And then you can use it:

```
image = URLInputFile(
    "https://www.python.org/static/community_logos/python-powered-h-140x182.png",
    filename="python-logo.png"
)
```

## 2.3.8 Global defaults

aiogram provides mechanism to set some global defaults for all requests to Telegram Bot API in your application using aiogram.client.default.DefaultBotProperties class.

There are some properties that can be set:

Default bot properties.

parse\_mode: str | None

Default parse mode for messages.

disable\_notification: bool | None

Sends the message silently. Users will receive a notification with no sound.

protect\_content: bool | None Protects content from copying.

allow\_sending\_without\_reply: bool | None

Allows to send messages without reply.

link\_preview: LinkPreviewOptions | None

Link preview settings.

link\_preview\_is\_disabled: bool | None

Disables link preview.

link\_preview\_prefer\_small\_media: bool | None

Prefer small media in link preview.

link\_preview\_prefer\_large\_media: bool | None

Prefer large media in link preview.

link\_preview\_show\_above\_text: bool | None

Show link preview above text.

show\_caption\_above\_media: bool | None

Show caption above media.



If you need to override default properties for some requests, you should use aiogram.client.default.DefaultBotProperties only for properties that you want to set as defaults and pass explicit values for other properties.

# **Danger**

If you upgrading from aiogram 3.0-3.6 to 3.7, you should update your code to use aiogram.client.default.DefaultBotProperties.

## **Example**

Here is an example of setting default parse mode for all requests to Telegram Bot API:

```
bot = Bot(
    token=...,
    default=DefaultBotProperties(
        parse_mode=ParseMode.HTML,
    )
)
```

In this case all messages sent by this bot will be parsed as HTML, so you don't need to specify *parse\_mode* in every message you send.

Instead of

```
await bot.send_message(chat_id, text, parse_mode=ParseMode.HTML)
```

you can use

```
await bot.send_message(chat_id, text)
```

and the message will be sent with HTML parse mode.

In some cases you may want to override default properties for some requests. You can do it by passing explicit values to the method:

```
await bot.send_message(chat_id, text, parse_mode=ParseMode.MARKDOWN_V2)
```

In this case the message will be sent with Markdown parse mode instead of default HTML.

Another example of overriding default properties:

```
await bot.send_message(chat_id, text, parse_mode=None)
```

In this case the message will be send without parse mode, even if default parse mode is set it may be useful if you want to send message with plain text or aiogram.types.message\_entity.MessageEntity.

```
await bot.send_message(
    chat_id=chat_id,
    text=text,
    entities=[MessageEntity(type='bold', offset=0, length=4)],
    parse_mode=None
)
```

# 2.4 Handling events

aiogram includes Dispatcher mechanism. Dispatcher is needed for handling incoming updates from Telegram.

With dispatcher you can do:

- Handle incoming updates;
- Filter incoming events before it will be processed by specific handler;
- · Modify event and related data in middlewares;
- Separate bot functionality between different handlers, modules and packages

Dispatcher is also separated into two entities - Router and Dispatcher. Dispatcher is subclass of router and should be always is root router.

Telegram supports two ways of receiving updates:

- Webhook you should configure your web server to receive updates from Telegram;
- Long polling you should request updates from Telegram.

So, you can use both of them with aiogram.

# **2.4.1 Router**

Usage:

```
from aiogram import Router
from aiogram.types import Message

my_router = Router(name=__name__)

@my_router.message()
async def message_handler(message: Message) -> Any:
    await message.answer('Hello from my router!')
```

```
class aiogram.dispatcher.router.Router(*, name: str | None = None)
```

Bases: object

Router can route update, and it nested update types like messages, callback query, polls and all other event types.

Event handlers can be registered in observer by two ways:

```
    By observer method - router.<event_type>.register(handler, <filters, ...>)
    By decorator - @router.<event_type>(<filters, ...>)
    __init__(*, name: str | None = None) → None
```

#### **Parameters**

name - Optional router name, can be useful for debugging

```
include\_router(router: Router) \rightarrow Router
```

Attach another router.

Parameters router

Returns

```
include_routers(*routers: Router) → None
     Attach multiple routers.
         Parameters
             routers
         Returns
resolve\_used\_update\_types(skip\_events: Set[str] | None = None) \rightarrow List[str]
     Resolve registered event names
     Is useful for getting updates only for registered event types.
         Parameters
              skip_events – skip specified event names
         Returns
```

#### **Event observers**



#### Warning

All handlers always should be asynchronous. The name of the handler function is not important. The event argument name is also not important but it is recommended to not overlap the name with contextual data in due to function can not accept two arguments with the same name.

Here is the list of available observers and examples of how to register handlers

set of registered names

In these examples only decorator-style registering handlers are used, but if you don't like @decorators just use <event type>.register(...) method instead.

#### Message

#### Attention

Be attentive with filtering this event

You should expect that this event can be with different sets of attributes in different cases

(For example text, sticker and document are always of different content types of message)

Recommended way to check field availability before usage, for example via magic filter: F.text to handle text, F.sticker to handle stickers only and etc.

```
@router.message()
async def message_handler(message: types.Message) -> Any: pass
```

## **Edited message**

```
@router.edited_message()
async def edited_message_handler(edited_message: types.Message) -> Any: pass
```

# **Channel post**

```
@router.channel_post()
async def channel_post_handler(channel_post: types.Message) -> Any: pass
```

# **Edited channel post**

```
@router.edited_channel_post()
async def edited_channel_post_handler(edited_channel_post: types.Message) -> Any: pass
```

# **Inline query**

```
@router.inline_query()
async def inline_query_handler(inline_query: types.InlineQuery) -> Any: pass
```

# **Chosen inline query**

```
@router.chosen_inline_result()
async def chosen_inline_result_handler(chosen_inline_result: types.ChosenInlineResult) ->
      Any: pass
```

### Callback query

```
@router.callback_query()
async def callback_query_handler(callback_query: types.CallbackQuery) -> Any: pass
```

# **Shipping query**

```
@router.shipping_query()
async def shipping_query_handler(shipping_query: types.ShippingQuery) -> Any: pass
```

## Pre checkout query

### Poll

```
@router.poll()
async def poll_handler(poll: types.Poll) -> Any: pass
```

#### Poll answer

```
@router.poll_answer()
async def poll_answer_handler(poll_answer: types.PollAnswer) -> Any: pass
```

## My chat member

```
@router.my_chat_member()
async def my_chat_member_handler(my_chat_member: types.ChatMemberUpdated) -> Any: pass
```

### **Chat member**

```
@router.chat_member()
async def chat_member_handler(chat_member: types.ChatMemberUpdated) -> Any: pass
```

## Chat join request

```
@router.chat_join_request()
async def chat_join_request_handler(chat_join_request: types.ChatJoinRequest) -> Any:_
_pass
```

#### Message reaction

572

## Message reaction count

```
@router.message_reaction_count()
async def message_reaction_count_handler(message_reaction_count: types.
→MessageReactionCountUpdated) -> Any: pass
```

#### **Chat boost**

```
@router.chat_boost()
async def chat_boost_handler(chat_boost: types.ChatBoostUpdated) -> Any: pass
```

#### Remove chat boost

```
@router.removed_chat_boost()
async def removed_chat_boost_handler(removed_chat_boost: types.ChatBoostRemoved) -> Any:_
→pass
```

### **Errors**

```
@router.errors()
async def error_handler(exception: types.ErrorEvent) -> Any: pass
```

Is useful for handling errors from other handlers, error event described *here* 

#### **Nested routers**



### Warning

## Routers by the way can be nested to an another routers with some limitations:

1. Router CAN NOT include itself 1. Routers CAN NOT be used for circular including (router 1 include router 2, router 2 include router 3, router 3 include router 1)

Example:

```
Listing 1: module_1.py
```

#### name

```
module_1
router2 = Router()
@router2.message() ...
```

Listing 2: module\_2.py

#### name

 $module\_2$ 

from module\_2 import router2

router1 = Router() router1.include\_router(router2)

# **Update**

```
@dispatcher.update()
async def message_handler(update: types.Update) -> Any: pass
```

# ▲ Warning

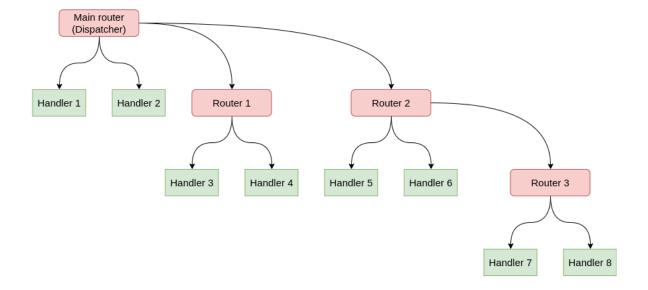
The only root Router (Dispatcher) can handle this type of event.

# 1 Note

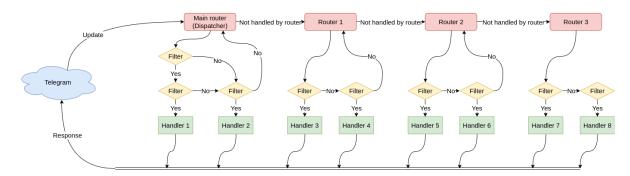
Dispatcher already has default handler for this event type, so you can use it for handling all updates that are not handled by any other handlers.

# How it works?

For example, dispatcher has 2 routers, the last router also has one nested router:



In this case update propagation flow will have form:



# 2.4.2 Dispatcher

Dispatcher is root *Router* and in code Dispatcher can be used directly for routing updates or attach another routers into dispatcher.

Here is only listed base information about Dispatcher. All about writing handlers, filters and etc. you can find in next pages:

- Router
- Filtering events

Root router

```
__init__(*, storage: BaseStorage | None = None, fsm_strategy: FSMStrategy = FSMStrategy.USER_IN_CHAT, events_isolation: BaseEventIsolation | None = None, disable_fsm: bool = False, name: str | None = None, **kwargs: Any) → None
```

Root router

#### **Parameters**

- storage Storage for FSM
- **fsm\_strategy** FSM strategy
- events\_isolation Events isolation
- **disable\_fsm** Disable FSM, note that if you disable FSM then you should not use storage and events isolation
- **kwargs** Other arguments, will be passed as keyword arguments to handlers

 $\textbf{async feed\_raw\_update}(\textit{bot: Bot, update: Dict[str, Any]}, **kwargs: Any) \rightarrow Any$ 

Main entry point for incoming updates with automatic Dict->Update serializer

### **Parameters**

- bot
- update

kwargs

```
async feed_update(bot: Bot, update: Update, **kwargs: Any) <math>\rightarrow Any
```

Main entry point for incoming updates Response of this method can be used as Webhook response

#### **Parameters**

- bot
- update

```
 \begin{aligned} \textbf{run\_polling}(*bots: Bot, polling\_timeout: int = 10, handle\_as\_tasks: bool = True, backoff\_config: \\ BackoffConfig = BackoffConfig(min\_delay=1.0, max\_delay=5.0, factor=1.3, jitter=0.1), \\ allowed\_updates: List[str] \mid \_SentinelObject \mid None = sentinel.UNSET, handle\_signals: bool \\ = True, close\_bot\_session: bool = True, **kwargs: Any) \rightarrow \text{None} \end{aligned}
```

Run many bots with polling

#### **Parameters**

- **bots** Bot instances (one or more)
- **polling\_timeout** Long-polling wait time
- handle\_as\_tasks Run task for each event and no wait result
- backoff\_config backoff-retry config
- allowed\_updates List of the update types you want your bot to receive
- handle\_signals handle signals (SIGINT/SIGTERM)
- close\_bot\_session close bot sessions on shutdown
- kwargs contextual data

# Returns

```
async start_polling(*bots: Bot, polling_timeout: int = 10, handle_as_tasks: bool = True, backoff_config: BackoffConfig = BackoffConfig(min_delay=1.0, max_delay=5.0, factor=1.3, jitter=0.1), allowed_updates: List[str] | _SentinelObject | None = sentinel.UNSET, handle_signals: bool = True, close_bot_session: bool = True, **kwargs: Any) \rightarrow None
```

### Polling runner

#### **Parameters**

- **bots** Bot instances (one or more)
- polling\_timeout Long-polling wait time
- handle\_as\_tasks Run task for each event and no wait result
- backoff\_config backoff-retry config
- **allowed\_updates** List of the update types you want your bot to receive By default, all used update types are enabled (resolved from handlers)
- handle\_signals handle signals (SIGINT/SIGTERM)
- close\_bot\_session close bot sessions on shutdown
- **kwargs** contextual data

#### Returns

```
async stop_polling() \rightarrow None
```

Execute this method if you want to stop polling programmatically

Returns

# Simple usage

Example:

```
dp = Dispatcher()

@dp.message()
async def message_handler(message: types.Message) -> None:
    await SendMessage(chat_id=message.from_user.id, text=message.text)
```

Including routers

Example:

```
dp = Dispatcher()
router1 = Router()
dp.include_router(router1)
```

# **Handling updates**

All updates can be propagated to the dispatcher by feed\_update() method:

```
from aiogram import Bot, Dispatcher

async def update_handler(update: Update, bot: Bot, dispatcher: Dispatcher):
    result = await dp.feed_update(bot, update)
```

Also you can feed raw update (dictionary) object to the dispatcher by feed\_raw\_update() method:

```
from aiogram import Bot, Dispatcher

async def update_handler(raw_update: dict[str, Any], bot: Bot, dispatcher: Dispatcher):
    result = await dp.feed_raw_update(bot, raw_update)
```

# 2.4.3 Dependency injection

Dependency injection is a programming technique that makes a class independent of its dependencies. It achieves that by decoupling the usage of an object from its creation. This helps you to follow SOLID's dependency inversion and single responsibility principles.

## How it works in aiogram

For each update aiogram.dispatcher.dispatcher.Dispatcher passes handling context data. Filters and middleware can also make changes to the context.

To access contextual data you should specify corresponding keyword parameter in handler or filter. For example, to get aiogram.fsm.context.FSMContext we do it like that:

```
@router.message(ProfileCompletion.add_photo, F.photo)
async def add_photo(
   message: types.Message, bot: Bot, state: FSMContext
) -> Any:
   ... # do something with photo
```

## Injecting own dependencies

Aiogram provides several ways to complement / modify contextual data.

The first and easiest way is to simply specify the named arguments in aiogram.dispatcher.dispatcher.Dispatcher initialization, polling start methods or aiogram.webhook.aiohttp\_server.SimpleRequestHandler initialization if you use webhooks.

```
async def main() -> None:
    dp = Dispatcher(..., foo=42)
    return await dp.start_polling(
        bot, bar="Bazz"
)
```

Analogy for webhook:

```
async def main() -> None:
    dp = Dispatcher(..., foo=42)
    handler = SimpleRequestHandler(dispatcher=dp, bot=bot, bar="Bazz")
    ... # starting webhook
```

<code>aiogram.dispatcher.Dispatcher</code>'s workflow data also can be supplemented by setting values as in a dictionary:

```
dp = Dispatcher(...)
dp["eggs"] = Spam()
```

The middlewares updates the context quite often. You can read more about them on this page:

Middlewares

The last way is to return a dictionary from the filter:

```
from typing import Any, Dict, Optional, Union

from aiogram import Router
from aiogram.filters import Filter
from aiogram.types import Message, User

router = Router(name=__name__)
```

```
class HelloFilter(Filter):
   def __init__(self, name: Optional[str] = None) -> None:
        self.name = name
    async def __call__(
        self,
       message: Message,
       event_from_user: User,
        # Filters also can accept keyword parameters like in handlers
   ) -> Union[bool, Dict[str, Any]]:
       if message.text.casefold() == "hello":
            # Returning a dictionary that will update the context data
            return {"name": event_from_user.mention_html(name=self.name)}
       return False
@router.message(HelloFilter())
async def my_handler(
   message: Message, name: str # Now we can accept "name" as named parameter
   return message.answer("Hello, {name}!".format(name=name))
```

...or using *MagicFilter* with .as\_(...) method.

# 2.4.4 Filtering events

Filters is needed for routing updates to the specific handler. Searching of handler is always stops on first match set of filters are pass. By default, all handlers has empty set of filters, so all updates will be passed to first handler that has empty set of filters.

aiogram has some builtin useful filters or you can write own filters.

#### **Builtin filters**

Here is list of builtin filters:

### Command

### **Usage**

- 1. Filter single variant of commands: Command("start")
- 2. Handle command by regexp pattern: Command(re.compile(r"item\_(\d+)"))
- 3. Match command by multiple variants: Command("item", re.compile(r"item\_(\d+)"))
- 4. Handle commands in public chats intended for other bots: Command("command", ignore\_mention=True)
- 5. Use aiogram.types.bot\_command.BotCommand object as command reference Command(BotCommand="command", description="My awesome command")

# Warning

Command cannot include spaces or any whitespace

This filter can be helpful for handling commands from the text messages.

Works only with aiogram.types.message.Message events which have the text.

```
__init__(*values: str | Pattern | BotCommand, commands: Sequence[str | Pattern | BotCommand] | str |
Pattern | BotCommand | None = None, prefix: str = '/', ignore_case: bool = False,
ignore_mention: bool = False, magic: MagicFilter | None = None)
```

List of commands (string or compiled regexp patterns)

#### **Parameters**

- **prefix** Prefix for command. Prefix is always a single char but here you can pass all of allowed prefixes, for example: "/!" will work with commands prefixed by "/" or "!".
- **ignore\_case** Ignore case (Does not work with regexp, use flags instead)
- **ignore\_mention** Ignore bot mention. By default, bot can not handle commands intended for other bots
- magic Validate command object via Magic filter after all checks done

When filter is passed the aiogram.filters.command.CommandObject will be passed to the handler argument command

```
class aiogram.filters.command.CommandObject(prefix: str = '/', command: str = '', mention: str \mid None = None, args: str \mid None = None, regexp_match: Match[str] \mid None = None, magic_result: Any | None = None)
```

Instance of this object is always has command and it prefix. Can be passed as keyword argument **command** to the handler

```
prefix: str = '/'
    Command prefix

command: str = ''
    Command without prefix and mention

mention: str | None = None
    Mention (if available)

args: str | None = None
    Command argument

regexp_match: Match[str] | None = None
    Will be presented match result if the command is presented as regexp in filter

magic_result: Any | None = None

property mentioned: bool
    This command has mention?
```

### property text: str

Generate original text from object

#### **Allowed handlers**

Allowed update types for this filter:

- message
- edited\_message

## ChatMemberUpdated

# **Usage**

Handle user leave or join events

```
from aiogram.filters import IS_MEMBER, IS_NOT_MEMBER

@router.chat_member(ChatMemberUpdatedFilter(IS_MEMBER >> IS_NOT_MEMBER))
async def on_user_leave(event: ChatMemberUpdated): ...

@router.chat_member(ChatMemberUpdatedFilter(IS_NOT_MEMBER >> IS_MEMBER))
async def on_user_join(event: ChatMemberUpdated): ...
```

Or construct your own terms via using pre-defined set of statuses and transitions.

# **Explanation**

### member\_status\_changed

You can import from aiogram. filters all available variants of statuses, status groups or transitions:

# **Statuses**

name	Description
CREATOR	Chat owner
ADMINISTRATOR	Chat administrator
MEMBER	Member of the chat
RESTRICTED	Restricted user (can be not member)
LEFT	Isn't member of the chat
KICKED	Kicked member by administrators

Statuses can be extended with *is\_member* flag by prefixing with + (for is\_member == True) or - (for is\_member == False) symbol, like +RESTRICTED or -RESTRICTED

# Status groups

The particular statuses can be combined via bitwise or operator, like CREATOR | ADMINISTRATOR

name	Description
IS_MEMBER	Combination of (CREATOR   ADMINISTRATOR   MEMBER   +RESTRICTED) statuses.
IS_ADMIN	Combination of (CREATOR   ADMINISTRATOR) statuses.
IS_NOT_MEMBER	Combination of (LEFT   KICKED   -RESTRICTED) statuses.

## **Transitions**

Transitions can be defined via bitwise shift operators >> and <<. Old chat member status should be defined in the left side for >> operator (right side for <<) and new status should be specified on the right side for >> operator (left side for

The direction of transition can be changed via bitwise inversion operator: ~JOIN\_TRANSITION will produce swap of old and new statuses.

name	Description	
JOIN_TRANSIT	Means status changed from IS_NOT_MEMBER to IS_MEMBER (IS_NOT_MEMBER >> IS_MEMBER)	
LEAVE_TRANSI	SI Means status changed from IS_MEMBER to IS_NOT_MEMBER (~JOIN_TRANSITION)	
PROMOTED_TRA	Means status changed from (MEMBER   RESTRICTED   LEFT   KICKED) >>	
	ADMINISTRATOR ((MEMBER   RESTRICTED   LEFT   KICKED) >> ADMINISTRATOR)	



Note that if you define the status unions (via |) you will need to add brackets for the statement before use shift operator in due to operator priorities.

### **Allowed handlers**

Allowed update types for this filter:

- my\_chat\_member
- chat\_member

## **Magic filters**



# 1 Note

This page still in progress. Has many incorrectly worded sentences.

Is external package maintained by aiogram core team.

By default installs with aiogram and also is available on PyPi - magic-filter. That's mean you can install it and use with any other libraries and in own projects without depending aiogram installed.

### Usage

The **magic\_filter** package implements class shortly named magic\_filter.F that's mean F can be imported from aiogram or magic\_filter. F is alias for MagicFilter.



Note that *aiogram* has an small extension over magic-filter and if you want to use this extension you should import magic from *aiogram* instead of *magic\_filter* package

The MagicFilter object is callable, supports *some actions* and memorize the attributes chain and the action which should be checked on demand.

So that's mean you can chain attribute getters, describe simple data validations and then call the resulted object passing single object as argument, for example make attributes chain F.foo.bar.baz then add action 'F.foo.bar.baz == 'spam' and then call the resulted object - (F.foo.bar.baz == 'spam').resolve(obj)

#### Possible actions

Magic filter object supports some of basic logical operations over object attributes

#### **Exists or not None**

Default actions.

```
F.photo # lambda message: message.photo
```

# **Equals**

```
F.text == 'hello' # lambda message: message.text == 'hello'
F.from_user.id == 42 # lambda message: message.from_user.id == 42
F.text != 'spam' # lambda message: message.text != 'spam'
```

### Is one of

Can be used as method named in\_ or as matmul operator @ with any iterable

```
F.from_user.id.in_({42, 1000, 123123}) # lambda query: query.from_user.id in {42, 1000, □ → 123123}
F.data.in_({'foo', 'bar', 'baz'}) # lambda query: query.data in {'foo', 'bar', 'baz'}
```

### **Contains**

```
F.text.contains('foo') # lambda message: 'foo' in message.text
```

# String startswith/endswith

Can be applied only for text attributes

```
F.text.startswith('foo') # lambda message: message.text.startswith('foo')
F.text.endswith('bar') # lambda message: message.text.startswith('bar')
```

## Regexp

```
F.text.regexp(r'Hello, .+') # lambda message: re.match(r'Hello, .+', message.text)
```

#### **Custom function**

Accepts any callable. Callback will be called when filter checks result

```
F.chat.func(lambda chat: chat.id == -42) # lambda message: (lambda chat: chat.id == -42) (message.chat)
```

### Inverting result

Any of available operation can be inverted by bitwise inversion - ~

```
~F.text # lambda message: not message.text
~F.text.startswith('spam') # lambda message: not message.text.startswith('spam')
```

### **Combining**

All operations can be combined via bitwise and/or operators - &/|

## Attribute modifiers - string manipulations

Make text upper- or lower-case

Can be used only with string attributes.

```
F.text.lower() == 'test' # lambda message: message.text.lower() == 'test'
F.text.upper().in_({'F00', 'BAR'}) # lambda message: message.text.upper() in {'F00', 'BAR'}
F.text.len() == 5 # lambda message: len(message.text) == 5
```

# Get filter result as handler argument

This part is not available in magic-filter directly but can be used with aiogram

```
from aiogram import F
...
@router.message(F.text.regexp(r"^(\d+)$").as_("digits"))
async def any_digits_handler(message: Message, digits: Match[str]):
    await message.answer(html.quote(str(digits)))
```

## Usage in aiogram

```
@router.message(F.text == 'hello')
@router.inline_query(F.data == 'button:1')
@router.message(F.text.startswith('foo'))
@router.message(F.content_type.in_({\text', 'sticker'}))
@router.message(F.text.regexp(r'\d+'))
...
# Many others cases when you will need to check any of available event attribute
```

### **MagicData**

### **Usage**

1. MagicData(F.event.from\_user.id == F.config.admin\_id) (Note that config should be passed from middleware)

### **Explanation**

```
class aiogram.filters.magic_data.MagicData(magic_data: MagicFilter)
    This filter helps to filter event with contextual data
```

### magic\_data

Can be imported:

• from aiogram.filters import MagicData

### **Allowed handlers**

Allowed update types for this filter:

- message
- edited\_message
- channel\_post
- edited\_channel\_post
- inline\_query
- chosen\_inline\_result
- callback\_query
- shipping\_query
- pre\_checkout\_query
- poll
- poll\_answer
- my\_chat\_member
- chat\_member
- chat\_join\_request
- error

## **Callback Data Factory & Filter**

### class aiogram.filters.callback\_data.CallbackData

Base class for callback data wrapper

This class should be used as super-class of user-defined callbacks.

The class-keyword prefix is required to define prefix and also the argument sep can be passed to define separator (default is :).

```
pack() \rightarrow str
```

Generate callback data string

### Returns

valid callback data for Telegram Bot API

```
classmethod unpack(value: str) → T
   Parse callback data string
   Parameters
       value – value from Telegram
   Returns
       instance of CallbackData

classmethod filter(rule: MagicFilter | None = None) → CallbackQueryFilter
   Generates a filter for callback query with rule
   Parameters
       rule – magic rule
   Returns
       instance of filter

model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.
```

# **Usage**

Create subclass of CallbackData:

```
class MyCallback(CallbackData, prefix="my"):
   foo: str
   bar: int
```

After that you can generate any callback based on this class, for example:

```
cb1 = MyCallback(foo="demo", bar=42)
cb1.pack() # returns 'my:demo:42'
cb1.unpack('my:demo:42') # returns <MyCallback(foo="demo", bar=42)>
```

So... Now you can use this class to generate any callbacks with defined structure

```
# Pass it into the markup
InlineKeyboardButton(
    text="demo",
    callback_data=MyCallback(foo="demo", bar="42").pack() # value should be packed to_
    string
)
...
```

... and handle by specific rules

```
# Filter callback by type and value of field :code:`foo`
@router.callback_query(MyCallback.filter(F.foo == "demo"))
async def my_callback_foo(query: CallbackQuery, callback_data: MyCallback):
    await query.answer(...)
    ...
    print("bar =", callback_data.bar)
```

Also can be used in *Keyboard builder*:

Another abstract example:

```
class Action(str, Enum):
   ban = "ban"
   kick = "kick"
   warn = "warn"
class AdminAction(CallbackData, prefix="adm"):
   action: Action
   chat_id: int
   user id: int
# Inside handler
builder = InlineKeyboardBuilder()
for action in Action:
   builder.button(
        text=action.value.title(),
        callback_data=AdminAction(action=action, chat_id=chat_id, user_id=user_id),
await bot.send_message(
   chat_id=admins_chat,
   text=f"What do you want to do with {html.quote(name)}",
   reply_markup=builder.as_markup(),
)
@router.callback_query(AdminAction.filter(F.action == Action.ban))
async def ban_user(query: CallbackQuery, callback_data: AdminAction, bot: Bot):
   await bot.ban_chat_member(
        chat_id=callback_data.chat_id,
       user_id=callback_data.user_id,
   )
```

### **Known limitations**

Allowed types and their subclasses:

- str
- int
- bool
- float
- Decimal (from decimal import Decimal)

- Fraction (from fractions import Fraction)
- UUID (from uuid import UUID)
- Enum (from enum import Enum, only for string enums)
- IntEnum (from enum import IntEnum, only for int enums)

# 1 Note

Note that the integer Enum's should be always is subclasses of IntEnum in due to parsing issues.

# **Exceptions**

This filters can be helpful for handling errors from the text messages.

```
\begin{cal}{class aiogram.filters.exception.} \textbf{ExceptionTypeFilter} (*exceptions: Type[Exception]) \\ Allows to match exception by type \end{cal}
```

exceptions

```
class aiogram.filters.exception.ExceptionMessageFilter(pattern: str | Pattern[str])
Allow to match exception by message
```

pattern

### **Allowed handlers**

Allowed update types for this filters:

• error

### Writing own filters

Filters can be:

- Asynchronous function (async def my\_filter(\*args, \*\*kwargs): pass)
- Synchronous function (def my\_filter(\*args, \*\*kwargs): pass)
- Anonymous function (lambda event: True)
- · Any awaitable object
- Subclass of aiogram.filters.base.Filter
- Instances of MagicFilter

and should return bool or dict. If the dictionary is passed as result of filter - resulted data will be propagated to the next filters and handler as keywords arguments.

### Base class for own filters

```
class aiogram.filters.base.Filter
```

If you want to register own filters like builtin filters you will need to write subclass of this class with overriding the \_\_call\_\_ method and adding filter attributes.

```
abstract async __call__(*args: Any, **kwargs: Any) → bool | Dict[str, Any]
```

This method should be overridden.

Accepts incoming event and should return boolean or dict.

#### Returns

bool or Dict[str, Any]

```
update_handler_flags(flags: Dict[str, Any]) \rightarrow None
```

Also if you want to extend handler flags with using this filter you should implement this method

#### **Parameters**

**flags** – existing flags, can be updated directly

## Own filter example

For example if you need to make simple text filter:

```
from aiogram import Router
from aiogram.filters import Filter
from aiogram.types import Message

router = Router()

class MyFilter(Filter):
    def __init__(self, my_text: str) -> None:
        self.my_text = my_text

    async def __call__(self, message: Message) -> bool:
        return message.text == self.my_text

@router.message(MyFilter("hello"))
async def my_handler(message: Message): ...
```

## **Combining Filters**

In general, all filters can be combined in two ways

## **Recommended way**

If you specify multiple filters in a row, it will be checked with an "and" condition:

```
@<router>.message(F.text.startswith("show"), F.text.endswith("example"))
```

Also, if you want to use two alternative ways to run the same handler ("or" condition) you can register the handler twice or more times as you like

```
@<router>.message(F.text == "hi")
@<router>.message(CommandStart())
```

Also sometimes you will need to invert the filter result, for example you have an *IsAdmin* filter and you want to check if the user is not an admin

```
@<router>.message(~IsAdmin())
```

# Another possible way

An alternative way is to combine using special functions  $(and_f(), or_f(), invert_f())$  from aiogram. filters module):

```
and_f(F.text.startswith("show"), F.text.endswith("example"))
or_f(F.text(text="hi"), CommandStart())
invert_f(IsAdmin())
and_f(<A>, or_f(<B>, <C>))
```

# 2.4.5 Long-polling

Long-polling is a technology that allows a Telegram server to send updates in case when you don't have dedicated IP address or port to receive webhooks for example on a developer machine.

To use long-polling mode you should use aiogram.dispatcher.dispatcher.Dispatcher.start\_polling() or aiogram.dispatcher.Dispatcher.run\_polling() methods.

# 1 Note

You can use polling from only one polling process per single Bot token, in other case Telegram server will return an error.

### 1 Note

If you will need to scale your bot, you should use webhooks instead of long-polling.

### 1 Note

If you will use multibot mode, you should use webhook mode for all bots.

### **Example**

This example will show you how to create simple echo bot based on long-polling.

```
import asyncio
import logging
import sys
from os import getenv
from aiogram import Bot, Dispatcher, html
from aiogram.client.default import DefaultBotProperties
from aiogram.enums import ParseMode
from aiogram.filters import CommandStart
from aiogram.types import Message
# Bot token can be obtained via https://t.me/BotFather
TOKEN = getenv("BOT_TOKEN")
# All handlers should be attached to the Router (or Dispatcher)
dp = Dispatcher()
@dp.message(CommandStart())
async def command_start_handler(message: Message) -> None:
    This handler receives messages with `/start` command
    # Most event objects have aliases for API methods that can be called in events'
# For example if you want to answer to incoming message you can use `message.answer(.
→...)` alias
    # and the target chat will be passed to :ref:`aiogram.methods.send_message.
→ SendMessage`
    # method automatically or call API method directly via
    # Bot instance: `bot.send_message(chat_id=message.chat.id, ...)`
   await message.answer(f"Hello, {html.bold(message.from_user.full_name)}!")
@dp.message()
async def echo_handler(message: Message) -> None:
   Handler will forward receive a message back to the sender
   By default, message handler will handle all message types (like a text, photo,
⇒sticker etc.)
    try:
        # Send a copy of the received message
       await message.send_copy(chat_id=message.chat.id)
   except TypeError:
        # But not all the types is supported to be copied so need to handle it
        await message.answer("Nice try!")
```

```
async def main() -> None:
    # Initialize Bot instance with default bot properties which will be passed to all_
    API calls
    bot = Bot(token=TOKEN, default=DefaultBotProperties(parse_mode=ParseMode.HTML))

# And the run events dispatching
    await dp.start_polling(bot)

if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, stream=sys.stdout)
    asyncio.run(main())
```

## 2.4.6 Finite State Machine

A finite-state machine (FSM) or finite-state automaton (FSA, plural: automata), finite automaton, or simply a state machine, is a mathematical model of computation.

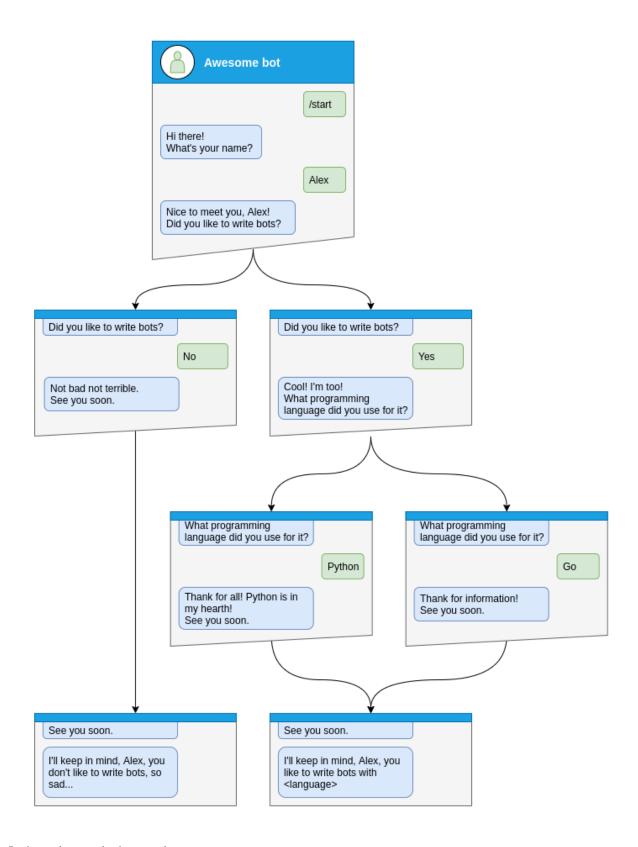
It is an abstract machine that can be in exactly one of a finite number of states at any given time. The FSM can change from one state to another in response to some inputs; the change from one state to another is called a transition.

An FSM is defined by a list of its states, its initial state, and the inputs that trigger each transition.

Source: WikiPedia

## Usage example

Not all functionality of the bot can be implemented as single handler, for example you will need to collect some data from user in separated steps you will need to use FSM.



Let's see how to do that step-by-step

594 Chapter 2. Contents

## Step by step

Before handle any states you will need to specify what kind of states you want to handle

```
class Form(StatesGroup):
   name = State()
   like_bots = State()
   language = State()
```

And then write handler for each state separately from the start of dialog

Here is dialog can be started only via command /start, so lets handle it and make transition user to state Form.name

```
@form_router.message(CommandStart())
async def command_start(message: Message, state: FSMContext) -> None:
    await state.set_state(Form.name)
    await message.answer(
        "Hi there! What's your name?",
        reply_markup=ReplyKeyboardRemove(),
    )
```

After that you will need to save some data to the storage and make transition to next step.

```
@form_router.message(Form.name)
async def process_name(message: Message, state: FSMContext) -> None:
    await state.update_data(name=message.text)
    await state.set_state(Form.like_bots)
   await message.answer(
        f"Nice to meet you, {html.quote(message.text)}!\nDid you like to write bots?",
        reply_markup=ReplyKeyboardMarkup(
            keyboard=[
                Ε
                    KeyboardButton(text="Yes"),
                    KeyboardButton(text="No"),
                ]
            ],
            resize_keyboard=True,
        ),
   )
```

At the next steps user can make different answers, it can be yes, no or any other

Handle yes and soon we need to handle Form.language state

```
@form_router.message(Form.like_bots, F.text.casefold() == "yes")
async def process_like_write_bots(message: Message, state: FSMContext) -> None:
    await state.set_state(Form.language)

await message.reply(
    "Cool! I'm too!\nWhat programming language did you use for it?",
    reply_markup=ReplyKeyboardRemove(),
)
```

Handle no

```
@form_router.message(Form.like_bots, F.text.casefold() == "no")
async def process_dont_like_write_bots(message: Message, state: FSMContext) -> None:
    data = await state.get_data()
    await state.clear()
    await message.answer(
        "Not bad not terrible.\nSee you soon.",
        reply_markup=ReplyKeyboardRemove(),
    )
    await show_summary(message=message, data=data, positive=False)
```

And handle any other answers

```
@form_router.message(Form.like_bots)
async def process_unknown_write_bots(message: Message) -> None:
    await message.reply("I don't understand you :(")
```

All possible cases of like\_bots step was covered, let's implement finally step

```
async def show_summary(message: Message, data: Dict[str, Any], positive: bool = True) ->_
None:
    name = data["name"]
    language = data.get("language", "<something unexpected>")
    text = f"I'll keep in mind that, {html.quote(name)}, "
    text += (
        f"you like to write bots with {html.quote(language)}."
        if positive
        else "you don't like to write bots, so sad..."
    )
    await message.answer(text=text, reply_markup=ReplyKeyboardRemove())
```

And now you have covered all steps from the image, but you can make possibility to cancel conversation, lets do that via command or text

```
@form_router.message(Command("cancel"))
@form_router.message(F.text.casefold() == "cancel")
async def cancel_handler(message: Message, state: FSMContext) -> None:
    """
    Allow user to cancel any action
    """
    current_state = await state.get_state()
    if current_state is None:
```

```
return

logging.info("Cancelling state %r", current_state)
await state.clear()
await message.answer(
    "Cancelled.",
    reply_markup=ReplyKeyboardRemove(),
)
```

# Complete example

```
import asyncio
   import logging
   import sys
   from os import getenv
   from typing import Any, Dict
   from aiogram import Bot, Dispatcher, F, Router, html
   from aiogram.client.default import DefaultBotProperties
   from aiogram.enums import ParseMode
   from aiogram.filters import Command, CommandStart
10
   from aiogram.fsm.context import FSMContext
11
   from aiogram.fsm.state import State, StatesGroup
12
   from aiogram.types import (
13
       KeyboardButton,
14
       Message,
       ReplyKeyboardMarkup,
16
       ReplyKeyboardRemove,
   )
18
   TOKEN = getenv("BOT_TOKEN")
20
21
   form_router = Router()
22
23
24
   class Form(StatesGroup):
25
       name = State()
26
       like_bots = State()
27
       language = State()
28
29
   @form_router.message(CommandStart())
31
   async def command_start(message: Message, state: FSMContext) -> None:
32
       await state.set_state(Form.name)
33
       await message.answer(
34
           "Hi there! What's your name?",
35
           reply_markup=ReplyKeyboardRemove(),
       )
37
38
```

```
@form_router.message(Command("cancel"))
40
   @form_router.message(F.text.casefold() == "cancel")
   async def cancel_handler(message: Message, state: FSMContext) -> None:
42
       Allow user to cancel any action
44
       current_state = await state.get_state()
46
       if current_state is None:
47
           return
48
49
       logging.info("Cancelling state %r", current_state)
50
       await state.clear()
51
       await message.answer(
52
            "Cancelled.",
53
           reply_markup=ReplyKeyboardRemove(),
       )
55
57
   @form_router.message(Form.name)
   async def process_name(message: Message, state: FSMContext) -> None:
59
       await state.update_data(name=message.text)
       await state.set state(Form.like bots)
61
       await message.answer(
62
           f"Nice to meet you, {html.quote(message.text)}!\nDid you like to write bots?",
63
           reply_markup=ReplyKeyboardMarkup(
                keyboard=[
65
                    Ε
                        KeyboardButton(text="Yes"),
                        KeyboardButton(text="No"),
68
                ],
70
                resize_keyboard=True,
           ),
72
       )
74
   @form_router.message(Form.like_bots, F.text.casefold() == "no")
76
   async def process_dont_like_write_bots(message: Message, state: FSMContext) -> None:
       data = await state.get_data()
78
       await state.clear()
       await message.answer(
80
           "Not bad not terrible.\nSee you soon.",
81
           reply_markup=ReplyKeyboardRemove(),
82
83
       await show_summary(message=message, data=data, positive=False)
84
85
   @form_router.message(Form.like_bots, F.text.casefold() == "yes")
87
   async def process_like_write_bots(message: Message, state: FSMContext) -> None:
       await state.set_state(Form.language)
89
       await message reply(
91
```

```
"Cool! I'm too!\nWhat programming language did you use for it?",
92
            reply_markup=ReplyKeyboardRemove(),
        )
    @form_router.message(Form.like_bots)
    async def process_unknown_write_bots(message: Message) -> None:
        await message.reply("I don't understand you :(")
100
101
   @form_router.message(Form.language)
102
    async def process_language(message: Message, state: FSMContext) -> None:
103
        data = await state.update_data(language=message.text)
        await state.clear()
105
        if message.text.casefold() == "python":
107
            await message.reply(
                "Python, you say? That's the language that makes my circuits light up! "
109
110
111
        await show_summary(message=message, data=data)
112
113
114
   async def show_summary(message: Message, data: Dict[str, Any], positive: bool = True) ->_
115
    None:
        name = data["name"]
116
        language = data.get("language", "<something unexpected>")
117
        text = f"I'll keep in mind that, {html.quote(name)}, "
118
        text += (
119
            f"you like to write bots with {html.quote(language)}."
120
            if positive
121
            else "you don't like to write bots, so sad..."
122
123
        await message.answer(text=text, reply_markup=ReplyKeyboardRemove())
125
    async def main():
127
        # Initialize Bot instance with default bot properties which will be passed to all
        bot = Bot(token=TOKEN, default=DefaultBotProperties(parse_mode=ParseMode.HTML))
129
130
        dp = Dispatcher()
131
132
        dp.include_router(form_router)
133
134
        # Start event dispatching
135
        await dp.start_polling(bot)
137
    if __name__ == "__main__":
139
        logging.basicConfig(level=logging.INFO, stream=sys.stdout)
140
        asyncio.run(main())
141
```

### **Read more**

### **Storages**

# Storages out of the box

# **MemoryStorage**

class aiogram.fsm.storage.memory.MemoryStorage

Default FSM storage, stores all data in dict and loss everything on shutdown



#### Warning

Is not recommended using in production in due to you will lose all data when your bot restarts

```
\_init\_() \rightarrow None
```

# RedisStorage

```
class aiogram.fsm.storage.redis.RedisStorage(redis: ~redis.asyncio.client.Redis, key_builder:
                                                         ~aiogram.fsm.storage.base.KeyBuilder | None = None,
                                                         state\_ttl: int \mid \sim datetime.timedelta \mid None = None,
                                                         data ttl: int | ~datetime.timedelta | None = None,
                                                         json_loads: ~typing.Callable[[...], ~typing.Any] =
                                                         <function loads>, json_dumps: ~typing.Callable[[...],
                                                         str = < function dumps>)
```

Redis storage required redis package installed (pip install redis)

```
__init__(redis: ~redis.asyncio.client.Redis, key_builder: ~aiogram.fsm.storage.base.KeyBuilder | None =
            None, state_ttl: int | ~datetime.timedelta | None = None, data_ttl: int | ~datetime.timedelta | None
            = None, json\_loads: \sim typing.Callable[[...], \sim typing.Any] = < function loads>, <math>json\_dumps:
            ~typing.Callable[[...], str] = <function dumps>) \rightarrow None
```

# **Parameters**

- **redis** Instance of Redis connection
- **key\_builder** builder that helps to convert contextual key to string
- state\_ttl TTL for state records
- data\_ttl TTL for data records

classmethod from\_url(url: str, connection\_kwargs:  $Dict[str, Any] \mid None = None, **kwargs: Any) \rightarrow$ RedisStorage

Create an instance of *RedisStorage* with specifying the connection string

### **Parameters**

- url for example redis://user:password@host:port/db
- connection\_kwargs see redis docs
- **kwargs** arguments to be passed to *RedisStorage*

#### Returns

an instance of RedisStorage

### MongoStorage

```
class aiogram.fsm.storage.mongo.MongoStorage(client: AsyncIOMotorClient, key_builder: KeyBuilder | None = None, db_name: str = 'aiogram\_fsm', collection\_name: str = 'states\_and\_data')
```

MongoDB storage required motor package installed (pip install motor)

```
__init__(client: AsyncIOMotorClient, key_builder: KeyBuilder | None = None, db_name: str = 'aiogram\_fsm', collection_name: str = 'states\_and\_data') \rightarrow None
```

#### **Parameters**

- **client** Instance of AsyncIOMotorClient
- **key\_builder** builder that helps to convert contextual key to string
- db\_name name of the MongoDB database for FSM
- collection\_name name of the collection for storing FSM states and data

classmethod from\_url(url: str,  $connection\_kwargs$ :  $Dict[str, Any] \mid None = None$ , \*\*kwargs: Any)  $\rightarrow$  MongoStorage

Create an instance of MongoStorage with specifying the connection string

#### **Parameters**

- url for example mongodb://user:password@host:port
- connection\_kwargs see motor docs
- **kwargs** arguments to be passed to *MongoStorage*

#### Returns

an instance of MongoStorage

### KeyBuilder

Keys inside Redis and Mongo storages can be customized via key builders:

```
class aiogram.fsm.storage.base.KeyBuilder
```

Base class for key builder.

**abstract build**(*key: StorageKey, part: Literal['data', 'state', 'lock']* | None = None)  $\rightarrow$  str Build key to be used in storage's db queries

### **Parameters**

- **key** contextual key
- part part of the record

# Returns

key to be used in storage's db queries

```
class aiogram.fsm.storage.base.DefaultKeyBuilder(*, prefix: str = 'fsm', separator: str = ':',
                                                              with bot id: bool = False,
                                                              with\_business\_connection\_id: bool = False,
                                                              with\_destiny: bool = False)
      Simple key builder with default prefix.
      Generates a colon-joined string with prefix, chat_id, user_id, optional bot_id, business_connection_id, destiny
      and field.
      Format:
           <prefix>:<bot_id?>:<business_connection_id?>:<chat_id>:<user_id>:<destiny?</pre>
           >:<field?>
      build(key: StorageKey, part: Literal['data', 'state', 'lock'] | None = None) \rightarrow str
           Build key to be used in storage's db queries
               Parameters
                    • key – contextual key
                    • part – part of the record
               Returns
                    key to be used in storage's db queries
Writing own storages
class aiogram.fsm.storage.base.BaseStorage
      Base class for all FSM storages
      abstract async set_state(key: StorageKey, state: str \mid State \mid None = None) \rightarrow None
           Set state for specified key
               Parameters
                    • key - storage key
                    • state - new state
      abstract async get_state(key: StorageKey) \rightarrow str | None
           Get key state
               Parameters
                   key - storage key
               Returns
                   current state
      abstract async set_data(key: StorageKey, data: Dict[str, Any]) \rightarrow None
           Write data (replace)
               Parameters
                    • key - storage key
                    • data - new data
      abstract async get_data(key: StorageKey) \rightarrow Dict[str, Any]
```

Get current data for key

#### **Parameters**

key – storage key

#### Returns

current data

**async update\_data**( $key: StorageKey, data: Dict[str, Any]) \rightarrow Dict[str, Any]$ 

Update date in the storage for key (like dict.update)

#### **Parameters**

- **key** storage key
- data partial data

#### **Returns**

new data

## abstract async close() $\rightarrow$ None

Close storage (database connection, file or etc.)

## **Strategy**

This module provides the FSMStrategy enumeration which is used to define the strategy of the finite state machine.

```
class aiogram.fsm.strategy.FSMStrategy(value, names=None, *, module=None, qualname=None,
                                           type=None, start=1, boundary=None)
```

FSM strategy for storage key generation.

#### CHAT = 2

State will be stored for each chat globally without separating by users.

# $CHAT_TOPIC = 5$

State will be stored for each chat and topic, but not separated by users.

#### $GLOBAL\_USER = 3$

State will be stored globally for each user globally.

### $USER_IN_CHAT = 1$

State will be stored for each user in chat.

## $USER_IN_TOPIC = 4$

State will be stored for each user in chat and topic.

# **Scenes Wizard**

Added in version 3.2.



### Warning

This feature is experimental and may be changed in future versions.

aiogram's basics API is easy to use and powerful, allowing the implementation of simple interactions such as triggering a command or message for a response. However, certain tasks require a dialogue between the user and the bot. This is where Scenes come into play.

## **Understanding Scenes**

A Scene in **aiogram** is like an abstract, isolated namespace or room that a user can be ushered into via the code. When a user is within a Scene, most other global commands or message handlers are bypassed, unless they are specifically designed to function outside of the Scenes. This helps in creating an experience of focused interactions. Scenes provide a structure for more complex interactions, effectively isolating and managing contexts for different stages of the conversation. They allow you to control and manage the flow of the conversation in a more organized manner.

# Scene Lifecycle

Each Scene can be "entered", "left" of "exited", allowing for clear transitions between different stages of the conversation. For instance, in a multi-step form filling interaction, each step could be a Scene - the bot guides the user from one Scene to the next as they provide the required information.

### **Scene Listeners**

Scenes have their own hooks which are command or message listeners that only act while the user is within the Scene. These hooks react to user actions while the user is 'inside' the Scene, providing the responses or actions appropriate for that context. When the user is ushered from one Scene to another, the actions and responses change accordingly as the user is now interacting with the set of listeners inside the new Scene. These 'Scene-specific' hooks or listeners, detached from the global listening context, allow for more streamlined and organized bot-user interactions.

#### Scene Interactions

Each Scene is like a self-contained world, with interactions defined within the scope of that Scene. As such, only the handlers defined within the specific Scene will react to user's input during the lifecycle of that Scene.

### **Scene Benefits**

Scenes can help manage more complex interaction workflows and enable more interactive and dynamic dialogs between the user and the bot. This offers great flexibility in handling multi-step interactions or conversations with the users.

### **How to use Scenes**

For example we have a quiz bot, which asks the user a series of questions and then displays the results.

Lets start with the data models, in this example simple data models are used to represent the questions and answers, in real life you would probably use a database to store the data.

Listing 3: Questions list

```
@dataclass
class Answer:
    """
    Represents an answer to a question.
    """
    text: str
    """The answer text"""
```

```
is_correct: bool = False
    """Indicates if the answer is correct"""
@dataclass
class Question:
    Class representing a quiz with a question and a list of answers.
   text: str
   """The question text"""
   answers: list[Answer]
   """List of answers"""
   correct_answer: str = field(init=False)
   def __post_init__(self):
       self.correct_answer = next(answer.text for answer in self.answers if answer.is_
# Fake data, in real application you should use a database or something else
QUESTIONS = [
   Question(
       text="What is the capital of France?",
        answers=[
            Answer("Paris", is_correct=True),
            Answer("London"),
           Answer("Berlin"),
           Answer("Madrid"),
       ],
   ),
   Question(
       text="What is the capital of Spain?",
        answers=[
            Answer("Paris"),
           Answer("London"),
           Answer("Berlin"),
            Answer("Madrid", is_correct=True),
       ],
   ),
   Question(
       text="What is the capital of Germany?",
        answers=[
            Answer("Paris"),
            Answer("London"),
            Answer("Berlin", is_correct=True),
            Answer("Madrid"),
       ],
   ),
    Question(
```

```
text="What is the capital of England?",
    answers=[
        Answer("Paris"),
        Answer("London", is_correct=True),
        Answer("Berlin"),
        Answer("Madrid"),
    ],
),
Question(
    text="What is the capital of Italy?",
    answers=[
        Answer("Paris"),
        Answer("London"),
        Answer("Berlin"),
        Answer("Rome", is_correct=True),
    ],
),
```

Then, we need to create a Scene class that will represent the quiz game scene:

# **1** Note

Keyword argument passed into class definition describes the scene name - is the same as state of the scene.

### Listing 4: Quiz Scene

```
class QuizScene(Scene, state="quiz"):
    """
    This class represents a scene for a quiz game.

It inherits from Scene class and is associated with the state "quiz".
    It handles the logic and flow of the quiz game.
    """
```

Also we need to define a handler that helps to start the quiz game:

# Listing 5: Start command handler

```
quiz_router = Router(name=__name__)
# Add handler that initializes the scene
quiz_router.message.register(QuizScene.as_handler(), Command("quiz"))
```

Once the scene is defined, we need to register it in the SceneRegistry:

### Listing 6: Registering the scene

Chapter 2. Contents

```
dispatcher.include_router(quiz_router)

# To use scenes, you should create a SceneRegistry and register your scenes there
scene_registry = SceneRegistry(dispatcher)
# ... and then register a scene in the registry
# by default, Scene will be mounted to the router that passed to the SceneRegistry,
# but you can specify the router explicitly using the `router` argument
scene_registry.add(QuizScene)

return dispatcher
```

So, now we can implement the quiz game logic, each question is sent to the user one by one, and the user's answer is checked at the end of all questions.

Now we need to write an entry point for the question handler:

Listing 7: Question handler entry point

```
@on.message.enter()
   async def on_enter(self, message: Message, state: FSMContext, step: int | None = 0) -
\rightarrow Any:
       Method triggered when the user enters the quiz scene.
       It displays the current question and answer options to the user.
       :param message:
       :param state:
       :param step: Scene argument, can be passed to the scene using the wizard
       :return:
       if not step:
           # This is the first step, so we should greet the user
           await message.answer("Welcome to the quiz!")
       try:
           quiz = QUESTIONS[step]
       except IndexError:
           # This error means that the question's list is over
           return await self.wizard.exit()
       markup = ReplyKeyboardBuilder()
       markup.add(*[KeyboardButton(text=answer.text) for answer in quiz.answers])
       if step > 0:
           markup.button(text=" Back")
       markup.button(text=" Exit")
       await state.update_data(step=step)
       return await message.answer(
           text=QUESTIONS[step].text,
           reply_markup=markup.adjust(2).as_markup(resize_keyboard=True),
       )
```

Once scene is entered, we should expect the user's answer, so we need to write a handler for it, this handler should expect the text message, save the answer and retake the question handler for the next question:

Listing 8: Answer handler

```
@on.message(F.text)
async def answer(self, message: Message, state: FSMContext) -> None:
    """
    Method triggered when the user selects an answer.

It stores the answer and proceeds to the next question.

:param message:
:param state:
:return:
"""

data = await state.get_data()
step = data["step"]
answers = data.get("answers", {})
answers[step] = message.text
await state.update_data(answers=answers)

await self.wizard.retake(step=step + 1)
```

When user answer with unknown message, we should expect the text message again:

Listing 9: Unknown message handler

```
@on.message()
  async def unknown_message(self, message: Message) → None:
    """
    Method triggered when the user sends a message that is not a command or an_
    →answer.

It asks the user to select an answer.

:param message: The message received from the user.
:return: None
    """
    await message.answer("Please select an answer.")
```

When all questions are answered, we should show the results to the user, as you can see in the code below, we use *await self.wizard.exit()* to exit from the scene when questions list is over in the *QuizScene.on\_enter* handler.

Thats means that we need to write an exit handler to show the results to the user:

Listing 10: Show results handler

```
@on.message.exit()
  async def on_exit(self, message: Message, state: FSMContext) → None:
    """
    Method triggered when the user exits the quiz scene.

It calculates the user's answers, displays the summary, and clears the stored.
    →answers.
```

```
:param message:
:param state:
:return:
data = await state.get_data()
answers = data.get("answers", {})
correct = 0
incorrect = 0
user_answers = []
for step, quiz in enumerate(QUESTIONS):
    answer = answers.get(step)
    is_correct = answer == quiz.correct_answer
    if is_correct:
        correct += 1
        icon = ""
    else:
        incorrect += 1
        icon = ""
    if answer is None:
        answer = "no answer"
    user_answers.append(f"{quiz.text} ({icon} {html.quote(answer)})")
content = as_list(
    as_section(
        Bold("Your answers:"),
        as_numbered_list(*user_answers),
    ),
    as_section(
        Bold("Summary:"),
        as_list(
            as_key_value("Correct", correct),
            as_key_value("Incorrect", incorrect),
        ),
    ),
)
await message.answer(**content.as_kwargs(), reply_markup=ReplyKeyboardRemove())
await state.set_data({})
```

Also we can implement a actions to exit from the quiz game or go back to the previous question:

#### Listing 11: Exit handler

```
@on.message(F.text == " Exit")
async def exit(self, message: Message) -> None:
    """
Method triggered when the user selects the "Exit" button.

It exits the quiz.
```

```
:param message:
:return:
"""
await self.wizard.exit()
```

Listing 12: Back handler

```
@on.message(F.text == " Back")
async def back(self, message: Message, state: FSMContext) -> None:
    """
    Method triggered when the user selects the "Back" button.

It allows the user to go back to the previous question.

:param message:
:param state:
:return:
"""
    data = await state.get_data()
    step = data["step"]

previous_step = step - 1
if previous_step < 0:
    # In case when the user tries to go back from the first question,
    # we just exit the quiz
    return await self.wizard.exit()
return await self.wizard.back(step=previous_step)</pre>
```

Now we can run the bot and test the quiz game:

Listing 13: Run the bot

```
async def main():
    dp = create_dispatcher()
    bot = Bot(token=TOKEN)
    await dp.start_polling(bot)

if __name__ == "__main__":
    # Alternatively, you can use aiogram-cli:
    # `aiogram run polling quiz_scene:create_dispatcher --log-level info --token BOT_
    TOKEN`
    logging.basicConfig(level=logging.INFO)
    asyncio.run(main())
```

Complete them all

Listing 14: Quiz Example

```
import asyncio
import logging
from dataclasses import dataclass, field

(continues on next page)
```

```
from os import getenv
from typing import Any
from aiogram import Bot, Dispatcher, F, Router, html
from aiogram.filters import Command
from aiogram.fsm.context import FSMContext
from aiogram.fsm.scene import Scene, SceneRegistry, ScenesManager, on
from aiogram.fsm.storage.memory import SimpleEventIsolation
from aiogram.types import KeyboardButton, Message, ReplyKeyboardRemove
from aiogram.utils.formatting import (
   Bold,
    as_key_value,
   as_list.
    as_numbered_list,
   as_section,
from aiogram.utils.keyboard import ReplyKeyboardBuilder
TOKEN = getenv("BOT_TOKEN")
@dataclass
class Answer:
   Represents an answer to a question.
   text: str
   """The answer text"""
   is_correct: bool = False
    """Indicates if the answer is correct"""
@dataclass
class Question:
    Class representing a quiz with a question and a list of answers.
   text: str
   """The question text"""
   answers: list[Answer]
   """List of answers"""
   correct_answer: str = field(init=False)
   def __post_init__(self):
        self.correct_answer = next(answer.text for answer in self.answers if answer.is_
# Fake data, in real application you should use a database or something else
```

```
QUESTIONS = \Gamma
    Question(
        text="What is the capital of France?",
        answers=[
            Answer("Paris", is_correct=True),
            Answer("London"),
            Answer("Berlin"),
            Answer("Madrid"),
        ],
    ),
    Question(
        text="What is the capital of Spain?",
        answers=[
            Answer("Paris"),
            Answer("London"),
            Answer("Berlin"),
            Answer("Madrid", is_correct=True),
        ],
    ),
    Question(
        text="What is the capital of Germany?",
        answers=[
            Answer("Paris"),
            Answer("London"),
            Answer("Berlin", is_correct=True),
            Answer("Madrid"),
        ],
    ),
    Question(
        text="What is the capital of England?",
        answers=[
            Answer("Paris"),
            Answer("London", is_correct=True),
            Answer("Berlin"),
            Answer("Madrid"),
        ],
    ),
    Question(
        text="What is the capital of Italy?",
        answers=[
            Answer("Paris"),
            Answer("London"),
            Answer("Berlin"),
            Answer("Rome", is_correct=True),
        ],
    ),
]
class QuizScene(Scene, state="quiz"):
    This class represents a scene for a quiz game.
```

```
It inherits from Scene class and is associated with the state "quiz".
   It handles the logic and flow of the quiz game.
   @on.message.enter()
   async def on_enter(self, message: Message, state: FSMContext, step: int | None = 0) -
\rightarrow Any:
       Method triggered when the user enters the quiz scene.
       It displays the current question and answer options to the user.
       :param message:
       :param state:
       :param step: Scene argument, can be passed to the scene using the wizard
       if not step:
           # This is the first step, so we should greet the user
           await message.answer("Welcome to the quiz!")
       try:
           quiz = QUESTIONS[step]
       except IndexError:
           # This error means that the question's list is over
           return await self.wizard.exit()
       markup = ReplyKeyboardBuilder()
       markup.add(*[KeyboardButton(text=answer.text) for answer in quiz.answers])
       if step > 0:
           markup.button(text=" Back")
       markup.button(text=" Exit")
       await state.update_data(step=step)
       return await message.answer(
           text=QUESTIONS[step].text,
           reply_markup=markup.adjust(2).as_markup(resize_keyboard=True),
       )
   @on.message.exit()
   async def on_exit(self, message: Message, state: FSMContext) -> None:
       Method triggered when the user exits the quiz scene.
       It calculates the user's answers, displays the summary, and clears the stored.
⊶answers.
       :param message:
       :param state:
       :return:
```

```
......
    data = await state.get_data()
    answers = data.get("answers", {})
    correct = 0
    incorrect = 0
    user_answers = []
    for step, quiz in enumerate(QUESTIONS):
        answer = answers.get(step)
        is_correct = answer == quiz.correct_answer
        if is_correct:
            correct += 1
            icon = ""
        else:
            incorrect += 1
            icon = ""
        if answer is None:
            answer = "no answer"
        user_answers.append(f"{quiz.text} ({icon} {html.quote(answer)})")
    content = as_list(
        as_section(
            Bold("Your answers:"),
            as_numbered_list(*user_answers),
        ),
        as_section(
            Bold("Summary:"),
            as_list(
                as_key_value("Correct", correct),
                as_key_value("Incorrect", incorrect),
            ),
        ),
    )
    await message.answer(**content.as_kwargs(), reply_markup=ReplyKeyboardRemove())
    await state.set_data({})
@on.message(F.text == " Back")
async def back(self, message: Message, state: FSMContext) -> None:
    Method triggered when the user selects the "Back" button.
    It allows the user to go back to the previous question.
    :param message:
    :param state:
    :return:
    data = await state.get_data()
    step = data["step"]
```

```
previous_step = step - 1
       if previous_step < 0:</pre>
           # In case when the user tries to go back from the first question,
           # we just exit the quiz
           return await self.wizard.exit()
       return await self.wizard.back(step=previous_step)
   @on.message(F.text == " Exit")
   async def exit(self, message: Message) -> None:
       Method triggered when the user selects the "Exit" button.
       It exits the quiz.
       :param message:
       :return:
       await self.wizard.exit()
   @on.message(F.text)
   async def answer(self, message: Message, state: FSMContext) -> None:
       Method triggered when the user selects an answer.
       It stores the answer and proceeds to the next question.
       :param message:
       :param state:
        :return:
       data = await state.get_data()
       step = data["step"]
       answers = data.get("answers", {})
       answers[step] = message.text
       await state.update_data(answers=answers)
       await self.wizard.retake(step=step + 1)
   @on.message()
   async def unknown_message(self, message: Message) -> None:
       Method triggered when the user sends a message that is not a command or an.
→answer.
       It asks the user to select an answer.
       :param message: The message received from the user.
       :return: None
       await message.answer("Please select an answer.")
```

```
quiz_router = Router(name=__name__)
# Add handler that initializes the scene
quiz_router.message.register(QuizScene.as_handler(), Command("quiz"))
@quiz_router.message(Command("start"))
async def command_start(message: Message, scenes: ScenesManager):
   await scenes.close()
   await message.answer(
        "Hi! This is a quiz bot. To start the quiz, use the /quiz command.",
       reply_markup=ReplyKeyboardRemove(),
   )
def create_dispatcher():
    # Event isolation is needed to correctly handle fast user responses
   dispatcher = Dispatcher(
        events_isolation=SimpleEventIsolation(),
   dispatcher.include_router(quiz_router)
   # To use scenes, you should create a SceneRegistry and register your scenes there
   scene_registry = SceneRegistry(dispatcher)
    # ... and then register a scene in the registry
    # by default, Scene will be mounted to the router that passed to the SceneRegistry,
    # but you can specify the router explicitly using the `router` argument
   scene_registry.add(QuizScene)
   return dispatcher
async def main():
   dp = create_dispatcher()
   bot = Bot(token=TOKEN)
    await dp.start_polling(bot)
if __name__ == "__main__":
    # Alternatively, you can use aiogram-cli:
    # `aiogram run polling quiz_scene:create_dispatcher --log-level info --token BOT_
→ TOKEN`
   logging.basicConfig(level=logging.INFO)
    asyncio.run(main())
```

## Components

- aiogram.fsm.scene.Scene represents a scene, contains handlers
- aiogram.fsm.scene.SceneRegistry container for all scenes in the bot, used to register scenes and resolve them by name
- aiogram.fsm.scene.ScenesManager manages scenes for each user, used to enter, leave and resolve current scene for user
- aiogram. fsm. scene. SceneConfig scene configuration, used to configure scene
- aiogram.fsm.scene.SceneWizard scene wizard, used to interact with user in scene from active scene handler
- Markers marker for scene handlers, used to mark scene handlers

class aiogram.fsm.scene.Scene(wizard: SceneWizard)

Represents a scene in a conversation flow.

A scene is a specific state in a conversation where certain actions can take place.

Each scene has a set of filters that determine when it should be triggered, and a set of handlers that define the actions to be executed when the scene is active.



This class is not meant to be used directly. Instead, it should be subclassed to define custom scenes.

classmethod add\_to\_router(router: Router)  $\rightarrow$  None

Adds the scene to the given router.

**Parameters** 

router

Returns

**classmethod** as\_handler(\*\*kwargs: Any) → Callable[[...], Any]

Create an entry point handler for the scene, can be used to simplify the handler that starts the scene.

```
>>> router.message.register(MyScene.as_handler(), Command("start"))
```

classmethod as\_router( $name: str \mid None = None$ )  $\rightarrow Router$ 

Returns the scene as a router.

Returns

new router

**class** aiogram.fsm.scene.**SceneRegistry**(router: Router, register on add: bool = True)

A class that represents a registry for scenes in a Telegram bot.

add(\*scenes: Type[Scene], router: Router | None = None)  $\rightarrow$  None

This method adds the specified scenes to the registry and optionally registers it to the router.

If a scene with the same state already exists in the registry, a SceneException is raised.

## 🛕 Warning

If the router is not specified, the scenes will not be registered to the router. You will need to include the scenes manually to the router or use the register method.

#### **Parameters**

- scenes A variable length parameter that accepts one or more types of scenes. These scenes are instances of the Scene class.
- router An optional parameter that specifies the router to which the scenes should be added.

#### Returns

None

```
get(scene: Type[Scene] \mid str \mid None) \rightarrow Type[Scene]
```

This method returns the registered Scene object for the specified scene. The scene parameter can be either a Scene object or a string representing the name of the scene. If a Scene object is provided, the state attribute of the SceneConfig object associated with the Scene object will be used as the scene name. If None or an invalid type is provided, a SceneException will be raised.

If the specified scene is not registered in the SceneRegistry object, a SceneException will be raised.

#### **Parameters**

**scene** – A Scene object or a string representing the name of the scene.

#### Returns

The registered Scene object corresponding to the given scene parameter.

```
register(*scenes: Type[Scene]) \rightarrow None
```

Registers one or more scenes to the SceneRegistry.

#### **Parameters**

**scenes** – One or more scene classes to register.

## Returns

None

**class** aiogram.fsm.scene.**ScenesManager**(registry: SceneRegistry, update\_type: str, event: TelegramObject, state: FSMContext, data: Dict[str, Any])

The ScenesManager class is responsible for managing scenes in an application. It provides methods for entering and exiting scenes, as well as retrieving the active scene.

```
async close(**kwargs: Any) \rightarrow None
```

Close method is used to exit the currently active scene in the ScenesManager.

**kwargs** – Additional keyword arguments passed to the scene's exit method.

#### Returns

None

**async enter**( $scene\_type$ : Type[Scene] | str | None,  $\_check\_active$ : bool = True, \*\*kwargs: Any)  $\rightarrow$  None Enters the specified scene.

## **Parameters**

• scene\_type – Optional Type[Scene] or str representing the scene type to enter.

- **\_check\_active** Optional bool indicating whether to check if there is an active scene to exit before entering the new scene. Defaults to True.
- **kwargs** Additional keyword arguments to pass to the scene's wizard.enter() method.

#### **Returns**

None

 $\textbf{class} \ \ \textbf{aiogram.fsm.scene.SceneConfig}(\textit{state: 'Optional[str]'}, \textit{handlers: 'List[HandlerContainer]', actions: 'Dict[SceneAction, Dict[str, CallableObject]]', reset_data_on_enter: 'Optional[bool]' = None, reset_history_on_enter: 'Optional[bool]' = None, callback_query_without_state: 'Optional[bool]' = None)$ 

actions: Dict[SceneAction, Dict[str, CallableObject]]

Scene actions

callback\_query\_without\_state: bool | None = None

Allow callback query without state

handlers: List[HandlerContainer]

Scene handlers

reset\_data\_on\_enter: bool | None = None

Reset scene data on enter

reset\_history\_on\_enter: bool | None = None

Reset scene history on enter

state: str | None

Scene state

A class that represents a wizard for managing scenes in a Telegram bot.

Instance of this class is passed to each scene as a parameter. So, you can use it to transition between scenes, get and set data, etc.

## **1** Note

This class is not meant to be used directly. Instead, it should be used as a parameter in the scene constructor.

**async back**(\*\*kwargs: Any)  $\rightarrow$  None

This method is used to go back to the previous scene.

**Parameters** 

**kwargs** – Keyword arguments that can be passed to the method.

Returns

None

 $\textbf{async clear\_data()} \rightarrow None$ 

Clears the data.

Returns

None

```
async enter(**kwargs: Any) \rightarrow None
```

Enter method is used to transition into a scene in the SceneWizard class. It sets the state, clears data and history if specified, and triggers entering event of the scene.

#### **Parameters**

**kwargs** – Additional keyword arguments.

#### Returns

None

```
async exit(**kwargs: Any) \rightarrow None
```

Exit the current scene and enter the default scene/state.

#### **Parameters**

**kwargs** – Additional keyword arguments.

#### Returns

None

```
async get_data() \rightarrow Dict[str, Any]
```

This method returns the data stored in the current state.

#### Returns

A dictionary containing the data stored in the scene state.

```
async goto(scene: Type[Scene] \mid str, **kwargs: Any) \rightarrow None
```

The *goto* method transitions to a new scene. It first calls the *leave* method to perform any necessary cleanup in the current scene, then calls the *enter* event to enter the specified scene.

#### **Parameters**

- **scene** The scene to transition to. Can be either a *Scene* instance or a string representing the scene.
- **kwargs** Additional keyword arguments to pass to the *enter* method of the scene manager.

#### Returns

None

```
async leave(\_with\_history: bool = True, **kwargs: Any) <math>\rightarrow None
```

Leaves the current scene. This method is used to exit a scene and transition to the next scene.

## **Parameters**

- \_with\_history Whether to include history in the snapshot. Defaults to True.
- **kwargs** Additional keyword arguments.

## **Returns**

None

```
async retake(**kwargs: Any) \rightarrow None
```

This method allows to re-enter the current scene.

#### **Parameters**

**kwargs** – Additional keyword arguments to pass to the scene.

## Returns

None

## **async** $set_data(data: Dict[str, Any]) \rightarrow None$

Sets custom data in the current state.

#### **Parameters**

**data** – A dictionary containing the custom data to be set in the current state.

#### Returns

None

**async update\_data**(*data*: *Dict*[*str*, *Any*] | *None* = *None*, \*\**kwargs*: *Any*) → Dict[str, Any]

This method updates the data stored in the current state

#### **Parameters**

- data Optional dictionary of data to update.
- **kwargs** Additional key-value pairs of data to update.

#### Returns

Dictionary of updated data

#### **Markers**

Markers are similar to the Router event registering mechanism, but they are used to mark scene handlers in the Scene class.

It can be imported from from aiogram.fsm.scene import on and should be used as decorator.

Allowed event types:

- · message
- · edited\_message
- · channel\_post
- edited\_channel\_post
- inline\_query
- · chosen\_inline\_result
- callback\_query
- · shipping\_query
- pre\_checkout\_query
- poll
- poll\_answer
- my\_chat\_member
- · chat\_member
- · chat\_join\_request

Each event type can be filtered in the same way as in the Router.

Also each event type can be marked as scene entry point, exit point or leave point.

If you want to mark the scene can be entered from message or inline query, you should use on.message or on. inline\_query marker:

```
class MyScene(Scene, name="my_scene"):
    @on.message.enter()
    async def on_enter(self, message: types.Message):
        pass

    @on.callback_query.enter()
    async def on_enter(self, callback_query: types.CallbackQuery):
        pass
```

Scene has only three points for transitions:

- enter point when user enters to the scene
- leave point when user leaves the scene and the enter another scene
- exit point when user exits from the scene

#### 2.4.7 Middlewares

aiogram provides powerful mechanism for customizing event handlers via middlewares.

Middlewares in bot framework seems like Middlewares mechanism in web-frameworks like aiohttp, fastapi, Django or etc.) with small difference - here is implemented two layers of middlewares (before and after filters).



Middleware is function that triggered on every event received from Telegram Bot API in many points on processing pipeline.

#### **Base theory**

As many books and other literature in internet says:

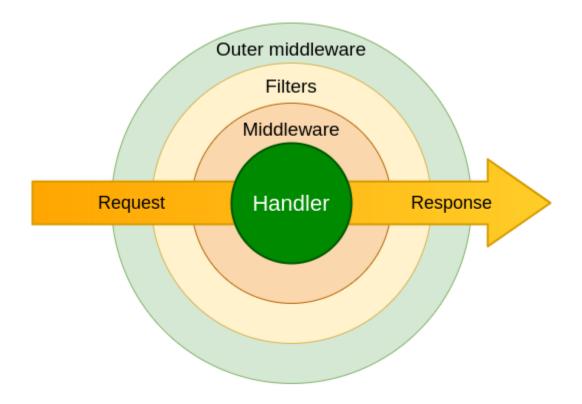
Middleware is reusable software that leverages patterns and frameworks to bridge the gap between the functional requirements of applications and the underlying operating systems, network protocol stacks, and databases.

Middleware can modify, extend or reject processing event in many places of pipeline.

#### **Basics**

Middleware instance can be applied for every type of Telegram Event (Update, Message, etc.) in two places

- 1. Outer scope before processing filters (<router>.<event>.outer\_middleware(...))
- 2. Inner scope after processing filters but before handler (<router>.<event>.middleware(...))



## **A** Attention

Middleware should be subclass of BaseMiddleware (from aiogram import BaseMiddleware) or any async callable

## **Arguments specification**

class aiogram.dispatcher.middlewares.base.BaseMiddleware

Bases: ABC

Generic middleware class

**abstract async \_\_call\_\_**(handler: Callable[[TelegramObject, Dict[str, Any]], Awaitable[Any]], event: TelegramObject, data: Dict[str, Any])  $\rightarrow$  Any

Execute middleware

#### **Parameters**

- handler Wrapped handler in middlewares chain
- event Incoming event (Subclass of aiogram.types.base.TelegramObject)
- data Contextual data. Will be mapped to handler arguments

## Returns

Any

## **Examples**

## **Danger**

Middleware should always call await handler(event, data) to propagate event for next middleware/handler. If you want to stop processing event in middleware you should not call await handler(event, data).

## Class-based

```
from aiogram import BaseMiddleware
from aiogram.types import Message

class CounterMiddleware(BaseMiddleware):
    def __init__(self) -> None:
        self.counter = 0

    async def __call__(
        self,
        handler: Callable[[Message, Dict[str, Any]], Awaitable[Any]],
        event: Message,
        data: Dict[str, Any]
) -> Any:
        self.counter += 1
        data['counter'] = self.counter
        return await handler(event, data)
```

and then

```
router = Router()
router.message.middleware(CounterMiddleware())
```

#### **Function-based**

```
@dispatcher.update.outer_middleware()
async def database_transaction_middleware(
   handler: Callable[[Update, Dict[str, Any]], Awaitable[Any]],
   event: Update,
   data: Dict[str, Any]
) -> Any:
   async with database.transaction():
    return await handler(event, data)
```

#### **Facts**

- 1. Middlewares from outer scope will be called on every incoming event
- 2. Middlewares from inner scope will be called only when filters pass
- 3. Inner middlewares is always calls for aiogram. types.update.Update event type in due to all incoming updates going to specific event type handler through built in update handler

## 2.4.8 Errors

## **Handling errors**

Is recommended way that you should use errors inside handlers using try-except block, but in common cases you can use global errors handler at router or dispatcher level.

If you specify errors handler for router - it will be used for all handlers inside this router.

If you specify errors handler for dispatcher - it will be used for all handlers inside all routers.

```
@router.error(ExceptionTypeFilter(MyCustomException), F.update.message.as_("message"))
async def handle_my_custom_exception(event: ErrorEvent, message: Message):
    # do something with error
    await message.answer("Oops, something went wrong!")

@router.error()
async def error_handler(event: ErrorEvent):
    logger.critical("Critical error caused by %s", event.exception, exc_info=True)
    # do something with error
    ...
```

#### **ErrorEvent**

Exception

## **Error types**

#### exception aiogram.exceptions.AiogramError

Base exception for all aiogram errors.

## exception aiogram.exceptions.DetailedAiogramError(message: str)

Base exception for all aiogram errors with detailed message.

#### exception aiogram.exceptions.CallbackAnswerException

Exception for callback answer.

## exception aiogram.exceptions.SceneException

Exception for scenes.

#### exception aiogram.exceptions.UnsupportedKeywordArgument(message: str)

Exception raised when a keyword argument is passed as filter.

## exception aiogram.exceptions.TelegramAPIError(method: TelegramMethod, message: str)

Base exception for all Telegram API errors.

## **exception** aiogram.exceptions.**TelegramNetworkError**(method: TelegramMethod, message: str)

Base exception for all Telegram network errors.

# 

Exception raised when flood control exceeds.

## 

Exception raised when chat has been migrated to a supergroup.

## $\textbf{exception} \ \ \textbf{aiogram.exceptions.} \textbf{TelegramBadRequest} (\textit{method: TelegramMethod, message: str})$

Exception raised when request is malformed.

## **exception** aiogram.exceptions.**TelegramNotFound**(method: TelegramMethod, message: str)

Exception raised when chat, message, user, etc. not found.

## **exception** aiogram.exceptions.**TelegramConflictError**(method: TelegramMethod, message: str)

Exception raised when bot token is already used by another application in polling mode.

# **exception** aiogram.exceptions.**TelegramUnauthorizedError**(method: TelegramMethod, message: str)

Exception raised when bot token is invalid.

## **exception** aiogram.exceptions.**TelegramForbiddenError**(method: TelegramMethod, message: str)

Exception raised when bot is kicked from chat or etc.

## exception aiogram.exceptions.TelegramServerError(method: TelegramMethod, message: str)

Exception raised when Telegram server returns 5xx error.

#### **exception** aiogram.exceptions.**RestartingTelegram**(method: TelegramMethod, message: str)

Exception raised when Telegram server is restarting.

It seems like this error is not used by Telegram anymore, but it's still here for backward compatibility.

## **Currently, you should expect that Telegram can raise RetryAfter (with timeout 5 seconds)**

error instead of this one.

**exception** aiogram.exceptions.**TelegramEntityTooLarge**(*method: TelegramMethod, message: str*) Exception raised when you are trying to send a file that is too large.

**exception** aiogram.exceptions.**ClientDecodeError**(*message: str, original: Exception, data: Any*) Exception raised when client can't decode response. (Malformed response, etc.)

## 2.4.9 Flags

Flags is a markers for handlers that can be used in *middlewares* or special *utilities* to make classification of the handlers. Flags can be added to the handler via *decorators*, *handlers registration* or filters.

#### Via decorators

For example mark handler with chat\_action flag

```
from aiogram import flags

@flags.chat_action
async def my_handler(message: Message)
```

Or just for rate-limit or something else

```
from aiogram import flags

@flags.rate_limit(rate=2, key="something")
async def my_handler(message: Message)
```

## Via handler registration method

```
@router.message(..., flags={'chat_action': 'typing', 'rate_limit': {'rate': 5}})
```

#### Via filters

```
class Command(Filter):
    ...

def update_handler_flags(self, flags: Dict[str, Any]) -> None:
    commands = flags.setdefault("commands", [])
    commands.append(self)
```

#### Use in middlewares

 $\textbf{aiogram.dispatcher.flags.check\_flags}(\textit{handler: HandlerObject} \mid \textit{Dict[str, Any], magic: MagicFilter)} \rightarrow \\ \textbf{Any}$ 

Check flags via magic filter

#### **Parameters**

- handler handler object or data
- magic instance of the magic

#### Returns

the result of magic filter check

aiogram.dispatcher.flags.extract\_flags(handler:  $HandlerObject \mid Dict[str, Any]$ )  $\rightarrow$  Dict[str, Any] Extract flags from handler or middleware context data

#### **Parameters**

handler – handler object or data

#### Returns

dictionary with all handler flags

```
aiogram.dispatcher.flags.get_flag(handler: HandlerObject | Dict[str, Any], name: str, *, default: Any | None = None) \rightarrow Any
```

Get flag by name

## **Parameters**

- handler handler object or data
- name name of the flag
- **default** default value (None)

#### Returns

value of the flag or default

#### **Example in middlewares**

```
async def my_middleware(handler, event, data):
    typing = get_flag(data, "typing") # Check that handler marked with `typing` flag
    if not typing:
        return await handler(event, data)

async with ChatActionSender.typing(chat_id=event.chat.id):
        return await handler(event, data)
```

#### Use in utilities

For example you can collect all registered commands with handler description and then it can be used for generating commands help

```
def collect_commands(router: Router) -> Generator[Tuple[Command, str], None, None]:
   for handler in router.message.handlers:
       if "commands" not in handler.flags: # ignore all handler without commands
            continue
        # the Command filter adds the flag with list of commands attached to the handler
       for command in handler.flags["commands"]:
            yield command, handler.callback.__doc__ or ""
   # Recursively extract commands from nested routers
   for sub_router in router.sub_routers:
       yield from collect_commands(sub_router)
```

## 2.4.10 Webhook

Telegram Bot API supports webhook. If you set webhook for your bot, Telegram will send updates to the specified url. You can use aiogram.methods.set\_webhook.SetWebhook() method to specify a url and receive incoming updates on it.



#### 1 Note

If you use webhook, you can't use long polling at the same time.

Before start i'll recommend you to read official Telegram's documentation about webhook

After you read it, you can start to read this section.

Generally to use webhook with aiogram you should use any async web framework. By out of the box aiogram has an aiohttp integration, so we'll use it.



#### 1 Note

You can use any async web framework you want, but you should write your own integration if you don't use aiohttp.

## aiohttp integration

Out of the box aiogram has aiohttp integration, so you can use it.

Here is available few ways to do it using different implementations of the webhook controller:

- aiogram.webhook.aiohttp\_server.BaseRequestHandler Abstract class for aiohttp webhook controller
- aiogram.webhook.aiohttp\_server.SimpleRequestHandler-Simple webhook controller, uses single Bot
- aiogram.webhook.aiohttp\_server.TokenBasedRequestHandler-Token based webhook controller, uses multiple Bot instances and tokens

You can use it as is or inherit from it and override some methods.

**\_\_init\_\_**(*dispatcher*: Dispatcher, *handle\_in\_background*: bool = False, \*\*data: Any)  $\rightarrow$  None Base handler that helps to handle incoming request from an aiohttp and propagate it to the Dispatcher

#### **Parameters**

- dispatcher instance of aiogram.dispatcher.dispatcher.Dispatcher
- handle\_in\_background immediately responds to the Telegram instead of a waiting end of a handler process

```
register(app: None, /, path: str, **kwargs: Any) \rightarrow None Register route and shutdown callback
```

#### **Parameters**

- app instance of aiohttp Application
- **path** route path
- kwargs

```
abstract async resolve\_bot(request: Request) \rightarrow Bot
```

This method should be implemented in subclasses of this class.

Resolve Bot instance from request.

# Parameters request

#### **Returns**

Bot instance

```
class aiogram.webhook.aiohttp_server.SimpleRequestHandler(dispatcher: Dispatcher, bot: Bot, handle_in_background: bool = True, secret_token: str \mid None = None, **data: Any)
```

```
__init__(dispatcher: Dispatcher, bot: Bot, handle_in_background: bool = True, secret_token: str \mid None = None, **data: Any) \rightarrow None
```

Handler for single Bot instance

#### **Parameters**

- dispatcher instance of aiogram.dispatcher.dispatcher.Dispatcher
- handle\_in\_background immediately responds to the Telegram instead of a waiting end of handler process
- **bot** instance of aiogram.client.bot.Bot

```
async close() \rightarrow None
```

Close bot session

```
register(app: None, /, path: str, **kwargs: Any) → None
```

Register route and shutdown callback

## **Parameters**

• app – instance of aiohttp Application

- path route path
- kwargs

```
async resolve_bot(request: Request) \rightarrow Bot
```

This method should be implemented in subclasses of this class.

Resolve Bot instance from request.

#### **Parameters**

request

#### Returns

Bot instance

class aiogram.webhook.aiohttp\_server.TokenBasedRequestHandler(dispatcher: Dispatcher,

handle\_in\_background: bool = True, bot\_settings: Dict[str, Any] | None = None, \*\*data: Any)

\_\_init\_\_(dispatcher: Dispatcher, handle\_in\_background: bool = True, bot\_settings: Dict[str, Any] | None = *None*, \*\**data: Any*)  $\rightarrow$  None

Handler that supports multiple bots the context will be resolved from path variable 'bot token'



## 1 Note

This handler is not recommended in due to token is available in URL and can be logged by reverse proxy server or other middleware.

## **Parameters**

- dispatcher instance of aiogram.dispatcher.dispatcher.Dispatcher
- handle\_in\_background immediately responds to the Telegram instead of a waiting end of handler process
- bot\_settings kwargs that will be passed to new Bot instance

**register**( $app: None, /, path: str, **kwargs: Any) \rightarrow None$ 

Validate path, register route and shutdown callback

#### **Parameters**

- app instance of aiohttp Application
- path route path
- kwargs

**async resolve\_bot**(request: Request)  $\rightarrow$  Bot

Get bot token from a path and create or get from cache Bot instance

#### **Parameters**

request

## **Returns**

## Security

Telegram supports two methods to verify incoming requests that they are from Telegram:

## Using a secret token

When you set webhook, you can specify a secret token and then use it to verify incoming requests.

## **Using IP filtering**

You can specify a list of IP addresses from which you expect incoming requests, and then use it to verify incoming requests.

It can be acy using firewall rules or nginx configuration or middleware on application level.

So, aiogram has an implementation of the IP filtering middleware for aiohttp.

```
\label{lem:aiogram.webhook.aiohttp_server.ip_filter_middleware(\it ip\_filter: IPFilter) $\rightarrow$ Callable[[Request, Callable[[Request], Awaitable[StreamResponse]]], Awaitable[Any]] $$
```

```
Parameters
ip_filter
```

Returns

```
class aiogram.webhook.security.IPFilter(ips: Sequence[str | IPv4Network | IPv4Address] | None = None)
    __init__(ips: Sequence[str | IPv4Network | IPv4Address] | None = None)
```

## **Examples**

## Behind reverse proxy

In this example we'll use aiohttp as web framework and nginx as reverse proxy.

```
This example shows how to use webhook on behind of any reverse proxy (nginx, traefik, ingress etc.)

import logging
import sys
from os import getenv

from aiohttp import web

from aiogram import Bot, Dispatcher, Router
from aiogram.client.default import DefaultBotProperties
from aiogram.enums import ParseMode
from aiogram.filters import CommandStart
from aiogram.types import Message
```

```
from aiogram.utils.markdown import hbold
from aiogram.webhook.aiohttp_server import SimpleRequestHandler, setup_application
# Bot token can be obtained via https://t.me/BotFather
TOKEN = getenv("BOT_TOKEN")
# Webserver settings
# bind localhost only to prevent any external access
WEB\_SERVER\_HOST = "127.0.0.1"
# Port for incoming request from reverse proxy. Should be any available port
WEB\_SERVER\_PORT = 8080
# Path to webhook route, on which Telegram will send requests
WEBHOOK_PATH = "/webhook"
# Secret key to validate requests from Telegram (optional)
WEBHOOK_SECRET = "my-secret"
# Base URL for webhook will be used to generate webhook URL for Telegram,
# in this example it is used public DNS with HTTPS support
BASE_WEBHOOK_URL = "https://aiogram.dev"
# All handlers should be attached to the Router (or Dispatcher)
router = Router()
@router.message(CommandStart())
async def command_start_handler(message: Message) -> None:
    This handler receives messages with `/start` command
    # Most event objects have aliases for API methods that can be called in events'
    # For example if you want to answer to incoming message you can use `message.answer(.
→...)` alias
    # and the target chat will be passed to :ref:`aiogram.methods.send_message.
→ SendMessage`
    # method automatically or call API method directly via
    # Bot instance: `bot.send_message(chat_id=message.chat.id, ...)`
    await message.answer(f"Hello, {hbold(message.from_user.full_name)}!")
@router.message()
async def echo_handler(message: Message) -> None:
    Handler will forward receive a message back to the sender
   By default, message handler will handle all message types (like text, photo, sticker.
⇔etc.)
   try:
        # Send a copy of the received message
        await message.send_copy(chat_id=message.chat.id)
    except TypeError:
```

```
# But not all the types is supported to be copied so need to handle it
        await message.answer("Nice try!")
async def on_startup(bot: Bot) -> None:
    # If you have a self-signed SSL certificate, then you will need to send a public
    # certificate to Telegram
   await bot.set_webhook(f"{BASE_WEBHOOK_URL}{WEBHOOK_PATH}", secret_token=WEBHOOK_
→SECRET)
def main() -> None:
    # Dispatcher is a root router
   dp = Dispatcher()
    # ... and all other routers should be attached to Dispatcher
   dp.include_router(router)
    # Register startup hook to initialize webhook
   dp.startup.register(on_startup)
    # Initialize Bot instance with default bot properties which will be passed to all
→ API calls
   bot = Bot(token=TOKEN, default=DefaultBotProperties(parse_mode=ParseMode.HTML))
   # Create aiohttp.web.Application instance
   app = web.Application()
   # Create an instance of request handler.
    # aiogram has few implementations for different cases of usage
    # In this example we use SimpleRequestHandler which is designed to handle simple.
   webhook_requests_handler = SimpleRequestHandler(
        dispatcher=dp.
       bot=bot.
        secret_token=WEBHOOK_SECRET,
    # Register webhook handler on application
   webhook_requests_handler.register(app, path=WEBHOOK_PATH)
   # Mount dispatcher startup and shutdown hooks to aiohttp application
   setup_application(app, dp, bot=bot)
    # And finally start webserver
   web.run_app(app, host=WEB_SERVER_HOST, port=WEB_SERVER_PORT)
if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, stream=sys.stdout)
   main()
```

When you use nginx as reverse proxy, you should set *proxy\_pass* to your aiohttp server address.

```
location /webhook {
    proxy_set_header Host $http_host;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_redirect off;
    proxy_buffering off;
    proxy_pass http://127.0.0.1:8080;
}
```

## Without reverse proxy (not recommended)

In case without using reverse proxy, you can use aiohttp's ssl context.

Also this example contains usage with self-signed certificate.

```
This example shows how to use webhook with SSL certificate.
import logging
import ssl
import sys
from os import getenv
from aiohttp import web
from aiogram import Bot, Dispatcher, Router
from aiogram.client.default import DefaultBotProperties
from aiogram.enums import ParseMode
from aiogram.filters import CommandStart
from aiogram.types import FSInputFile, Message
from aiogram.utils.markdown import hbold
from aiogram.webhook.aiohttp_server import SimpleRequestHandler, setup_application
# Bot token can be obtained via https://t.me/BotFather
TOKEN = getenv("BOT_TOKEN")
# Webserver settings
# bind localhost only to prevent any external access
WEB\_SERVER\_HOST = "127.0.0.1"
# Port for incoming request from reverse proxy. Should be any available port
WEB_SERVER_PORT = 8080
# Path to webhook route, on which Telegram will send requests
WEBHOOK_PATH = "/webhook"
# Secret key to validate requests from Telegram (optional)
WEBHOOK_SECRET = "my-secret"
# Base URL for webhook will be used to generate webhook URL for Telegram,
# in this example it is used public address with TLS support
BASE_WEBHOOK_URL = "https://aiogram.dev"
# Path to SSL certificate and private key for self-signed certificate.
WEBHOOK_SSL_CERT = "/path/to/cert.pem"
```

```
WEBHOOK_SSL_PRIV = "/path/to/private.key"
# All handlers should be attached to the Router (or Dispatcher)
router = Router()
@router.message(CommandStart())
async def command_start_handler(message: Message) -> None:
    This handler receives messages with `/start` command
    # Most event objects have aliases for API methods that can be called in events'.
# For example if you want to answer to incoming message you can use `message.answer(.
→...)` alias
   # and the target chat will be passed to :ref:`aiogram.methods.send_message.
→ SendMessage`
    # method automatically or call API method directly via
    # Bot instance: `bot.send_message(chat_id=message.chat.id, ...)`
    await message.answer(f"Hello, {hbold(message.from_user.full_name)}!")
@router.message()
async def echo_handler(message: Message) -> None:
   Handler will forward receive a message back to the sender
   By default, message handler will handle all message types (like text, photo, sticker.
⇔etc.)
    1111111
   try:
        # Send a copy of the received message
       await message.send_copy(chat_id=message.chat.id)
   except TypeError:
        # But not all the types is supported to be copied so need to handle it
        await message.answer("Nice try!")
async def on_startup(bot: Bot) -> None:
    # In case when you have a self-signed SSL certificate, you need to send the.
→certificate
    # itself to Telegram servers for validation purposes
    # (see https://core.telegram.org/bots/self-signed)
    # But if you have a valid SSL certificate, you SHOULD NOT send it to Telegram.
⇔servers.
    await bot.set_webhook(
        f"{BASE_WEBHOOK_URL}{WEBHOOK_PATH}",
        certificate=FSInputFile(WEBHOOK_SSL_CERT),
        secret_token=WEBHOOK_SECRET,
   )
```

```
def main() -> None:
    # Dispatcher is a root router
   dp = Dispatcher()
    # ... and all other routers should be attached to Dispatcher
   dp.include_router(router)
    # Register startup hook to initialize webhook
   dp.startup.register(on_startup)
    # Initialize Bot instance with default bot properties which will be passed to all.
→ API calls
   bot = Bot(token=TOKEN, default=DefaultBotProperties(parse_mode=ParseMode.HTML))
    # Create aiohttp.web.Application instance
   app = web.Application()
   # Create an instance of request handler,
    # aiogram has few implementations for different cases of usage
    # In this example we use SimpleRequestHandler which is designed to handle simple.
   webhook_requests_handler = SimpleRequestHandler(
        dispatcher=dp,
       bot=bot,
       secret_token=WEBHOOK_SECRET,
    # Register webhook handler on application
   webhook_requests_handler.register(app, path=WEBHOOK_PATH)
    # Mount dispatcher startup and shutdown hooks to aiohttp application
   setup_application(app, dp, bot=bot)
    # Generate SSL context
    context = ssl.SSLContext(ssl.PROTOCOL_TLSv1_2)
   context.load_cert_chain(WEBHOOK_SSL_CERT, WEBHOOK_SSL_PRIV)
    # And finally start webserver
   web.run_app(app, host=WEB_SERVER_HOST, port=WEB_SERVER_PORT, ssl_context=context)
if __name__ == "__main__":
   logging.basicConfig(level=logging.INFO, stream=sys.stdout)
   main()
```

## With using other web framework

You can pass incoming request to aiogram's webhook controller from any web framework you want.

Read more about it in aiogram.dispatcher.dispatcher.Dispatcher.feed\_webhook\_update() or aiogram. dispatcher.dispatcher.Dispatcher.feed\_update() methods.

```
update = Update.model_validate(await request.json(), context={"bot": bot})
await dispatcher.feed_update(update)
```



If you want to use reply into webhook, you should check that result of the feed\_update methods is an instance of API method and build multipart/form-data or application/json response body manually.

#### 2.4.11 Class based handlers

A handler is a async callable which takes a event with contextual data and returns a response.

In aiogram it can be more than just an async function, these allow you to use classes which can be used as Telegram event handlers to structure your event handlers and reuse code by harnessing inheritance and mixins.

There are some base class based handlers what you need to use in your own handlers:

#### **BaseHandler**

Base handler is generic abstract class and should be used in all other class-based handlers.

Import: from aiogram.handlers import BaseHandler

By default you will need to override only method async def handle(self) -> Any:

This class also has a default initializer and you don't need to change it. The initializer accepts the incoming event and all contextual data, which can be accessed from the handler through attributes: event: TelegramEvent and data: Dict[Any, str]

If an instance of the bot is specified in context data or current context it can be accessed through bot class attribute.

## **Example**

```
class MyHandler(BaseHandler[Message]):
   async def handle(self) -> Any:
         await self.event.answer("Hello!")
```

## CallbackQueryHandler

```
class aiogram.handlers.callback_query.CallbackQueryHandler(event: T, **kwargs: Any)

There is base class for callback query handlers.
```

#### **Example:**

```
from aiogram.handlers import CallbackQueryHandler
...
@router.callback_query()
class MyHandler(CallbackQueryHandler):
    async def handle(self) -> Any: ...
```

```
Is alias for event.from_user

Is alias for event.from_user

property message: MaybeInaccessibleMessage | None
    Is alias for event.message

property callback_data: str | None
    Is alias for event.data
```

#### ChosenInlineResultHandler

There is base class for chosen inline result handlers.

## Simple usage

```
from aiogram.handlers import ChosenInlineResultHandler
...
@router.chosen_inline_result()
class MyHandler(ChosenInlineResultHandler):
    async def handle(self) -> Any: ...
```

## **Extension**

This base handler is subclass of *BaseHandler* with some extensions:

- self.chat is alias for self.event.chat
- self.from\_user is alias for self.event.from\_user

## **ErrorHandler**

There is base class for error handlers.

## Simple usage

```
from aiogram.handlers import ErrorHandler

...
@router.errors()
class MyHandler(ErrorHandler):
    async def handle(self) -> Any:
    log.exception(
        "Cause unexpected exception %s: %s",
        self.exception_name,
        self.exception_message
    )
```

#### **Extension**

This base handler is subclass of *BaseHandler* with some extensions:

- self.exception\_name is alias for self.event.\_\_class\_\_.\_\_name\_\_
- self.exception\_message is alias for str(self.event)

## InlineQueryHandler

There is base class for inline query handlers.

## Simple usage

```
from aiogram.handlers import InlineQueryHandler
...
@router.inline_query()
class MyHandler(InlineQueryHandler):
    async def handle(self) -> Any: ...
```

## **Extension**

This base handler is subclass of *BaseHandler* with some extensions:

- self.chat is alias for self.event.chat
- self.query is alias for self.event.query

### MessageHandler

There is base class for message handlers.

## Simple usage

```
from aiogram.handlers import MessageHandler
...
@router.message()
class MyHandler(MessageHandler):
    async def handle(self) -> Any:
        return SendMessage(chat_id=self.chat.id, text="PASS")
```

#### **Extension**

This base handler is subclass of *BaseHandler* with some extensions:

- self.chat is alias for self.event.chat
- self.from\_user is alias for self.event.from\_user

## **PollHandler**

There is base class for poll handlers.

#### Simple usage

```
from aiogram.handlers import PollHandler
...
@router.poll()
class MyHandler(PollHandler):
    async def handle(self) -> Any: ...
```

## **Extension**

This base handler is subclass of *BaseHandler* with some extensions:

- self.question is alias for self.event.question
- self.options is alias for self.event.options

## **PreCheckoutQueryHandler**

There is base class for callback query handlers.

## Simple usage

```
from aiogram.handlers import PreCheckoutQueryHandler
...
@router.pre_checkout_query()
class MyHandler(PreCheckoutQueryHandler):
    async def handle(self) -> Any: ...
```

#### **Extension**

This base handler is subclass of *BaseHandler* with some extensions:

• self.from\_user is alias for self.event.from\_user

## **ShippingQueryHandler**

There is base class for callback query handlers.

## Simple usage

```
from aiogram.handlers import ShippingQueryHandler
...
@router.shipping_query()
class MyHandler(ShippingQueryHandler):
    async def handle(self) -> Any: ...
```

## **Extension**

This base handler is subclass of *BaseHandler* with some extensions:

• self.from\_user is alias for self.event.from\_user

#### **ChatMemberHandler**

There is base class for chat member updated events.

#### Simple usage

```
from aiogram.handlers import ChatMemberHandler
...
@router.chat_member()
@router.my_chat_member()
class MyHandler(ChatMemberHandler):
    async def handle(self) -> Any: ...
```

#### **Extension**

This base handler is subclass of *BaseHandler* with some extensions:

• self.chat is alias for self.event.chat

# 2.5 Utils

# 2.5.1 Keyboard builder

Keyboard builder helps to dynamically generate markup.



Note that if you have static markup, it's best to define it explicitly rather than using builder, but if you have dynamic markup configuration, feel free to use builder as you wish.

#### Usage example

For example you want to generate inline keyboard with 10 buttons

```
builder = InlineKeyboardBuilder()
for index in range(1, 11):
   builder.button(text=f"Set {index}", callback_data=f"set:{index}")
```

then adjust this buttons to some grid, for example first line will have 3 buttons, the next lines will have 2 buttons

```
builder.adjust(3, 2)
```

also you can attach another builder to this one

```
another_builder = InlineKeyboardBuilder(...)... # Another builder with some buttons
builder.attach(another_builder)
```

or you can attach some already generated markup

```
markup = InlineKeyboardMarkup(inline_keyboard=[...]) # Some markup
builder.attach(InlineKeyboardBuilder.from_markup(markup))
```

and finally you can export this markup to use it in your message

```
await message.answer("Some text here", reply_markup=builder.as_markup())
```

Reply keyboard builder has the same interface

#### Warning

Note that you can't attach reply keyboard builder to inline keyboard builder and vice versa

## **Inline Keyboard**

class aiogram.utils.keyboard.InlineKeyboardBuilder(markup: List[List[InlineKeyboardButton]] | None = None)

Inline keyboard builder inherits all methods from generic builder

```
button(text: str, url: str | None = None, login_url: LoginUrl | None = None, callback_data: str |
         CallbackData | None = None, switch_inline_query: str | None = None,
         switch_inline_query_current_chat: str | None = None, callback_game: CallbackGame | None =
         None, pay: bool | None = None, **kwargs: Any) \rightarrow aiogram.utils.keyboard.InlineKeyboardBuilder
```

Add new inline button to markup

```
as_markup() → aiogram.types.inline_keyboard_markup.InlineKeyboardMarkup
```

Construct an InlineKeyboardMarkup

```
__init__(markup: List[List[InlineKeyboardButton]] | None = None) <math>\rightarrow None
```

**add**(\**buttons: ButtonType*) → KeyboardBuilder[ButtonType]

Add one or many buttons to markup.

# Parameters buttons

#### Returns

```
adjust(*sizes: int, repeat: bool = False) \rightarrow KeyboardBuilder[ButtonType]
```

Adjust previously added buttons to specific row sizes.

By default, when the sum of passed sizes is lower than buttons count the last one size will be used for tail of the markup. If repeat=True is passed - all sizes will be cycled when available more buttons count than all sizes

#### **Parameters**

- sizes
- repeat

#### Returns

```
property buttons: Generator[ButtonType, None, None]
```

Get flatten set of all buttons

#### Returns

```
copy() \rightarrow InlineKeyboardBuilder
```

Make full copy of current builder with markup

#### Returns

```
export() → List[List[ButtonType]]
```

Export configured markup as list of lists of buttons

```
>>> builder = KeyboardBuilder(button_type=InlineKeyboardButton)
>>> ... # Add buttons to builder
>>> markup = InlineKeyboardMarkup(inline_keyboard=builder.export())
```

#### Returns

#### classmethod from\_markup(markup: InlineKeyboardMarkup) $\rightarrow$ InlineKeyboardBuilder

Create builder from existing markup

# Parameters markup

#### Returns

```
row(*buttons: ButtonType, width: int | None = None) → KeyboardBuilder[ButtonType]
```

Add row to markup

When too much buttons is passed it will be separated to many rows

#### **Parameters**

- buttons
- width

#### Returns

#### Reply Keyboard

```
class aiogram.utils.keyboard.ReplyKeyboardBuilder(markup: List[KeyboardButton]] | None =
     Reply keyboard builder inherits all methods from generic builder
     button(text: str, request_contact: bool | None = None, request_location: bool | None = None, request_poll:
              KeyboardButtonPollType | None = None, **kwargs: Any) \rightarrow
              aiogram.utils.keyboard.ReplyKeyboardBuilder
           Add new button to markup
     as_markup() → aiogram.types.reply_keyboard_markup.ReplyKeyboardMarkup
           Construct an ReplyKeyboardMarkup
     __init__(markup: List[List[KeyboardButton]] | None = None) <math>\rightarrow None
     add(*buttons: ButtonType) → KeyboardBuilder[ButtonType]
           Add one or many buttons to markup.
               Parameters
                   buttons
               Returns
     adjust(*sizes: int, repeat: bool = False) \rightarrow KeyboardBuilder[ButtonType]
           Adjust previously added buttons to specific row sizes.
           By default, when the sum of passed sizes is lower than buttons count the last one size will be used for tail
           of the markup. If repeat=True is passed - all sizes will be cycled when available more buttons count than
           all sizes
               Parameters
                   sizes

    repeat

               Returns
     property buttons: Generator[ButtonType, None, None]
           Get flatten set of all buttons
               Returns
     copy() \rightarrow ReplyKeyboardBuilder
           Make full copy of current builder with markup
               Returns
     export() → List[List[ButtonType]]
           Export configured markup as list of lists of buttons
           >>> builder = KeyboardBuilder(button_type=InlineKeyboardButton)
           >>> ... # Add buttons to builder
           >>> markup = InlineKeyboardMarkup(inline_keyboard=builder.export())
```

#### Returns

```
classmethod from_markup(markup: ReplyKeyboardMarkup) → ReplyKeyboardBuilder
```

Create builder from existing markup

Parameters markup

Returns

```
row(*buttons: ButtonType, width: int | None = None) \rightarrow KeyboardBuilder[ButtonType] Add row to markup
```

When too much buttons is passed it will be separated to many rows

#### **Parameters**

- buttons
- · width

**Returns** 

# 2.5.2 Translation

In order to make you bot translatable you have to add a minimal number of hooks to your Python code.

These hooks are called translation strings.

The aiogram translation utils is build on top of GNU gettext Python module and Babel library.

#### Installation

Babel is required to make simple way to extract translation strings from your code

Can be installed from pip directly:

```
pip install Babel
```

or as aiogram extra dependency:

```
pip install aiogram[i18n]
```

#### Make messages translatable

In order to gettext need to know what the strings should be translated you will need to write translation strings.

For example:

# **\text{\text{Danger}}**

f-strings can't be used as translations string because any dynamic variables should be added to message after getting translated message

Also if you want to use translated string in keyword- or magic- filters you will need to use lazy gettext calls:

```
from aiogram import F
from aiogram.utils.i18n import lazy_gettext as __

@router.message(F.text == __("My menu entry"))
...
```

## **Danger**

Lazy gettext calls should always be used when the current language is not know at the moment

# **\dagger** Danger

Lazy gettext can't be used as value for API methods or any Telegram Object (like aiogram.types.inline\_keyboard\_button.InlineKeyboardButton or etc.)

#### Working with plural forms

The *gettext* from *aiogram.utils.i18n* is the one alias for two functions \_gettext\_ and \_ngettext\_ of GNU gettext Python module. Therefore, the wrapper for message strings is the same \_(). You need to pass three parameters to the function: a singular string, a plural string, and a value.

## **Configuring engine**

After you messages is already done to use gettext your bot should know how to detect user language

On top of your application the instance of aiogram.utils.i18n.I18n should be created

```
i18n = I18n(path="locales", default_locale="en", domain="messages")
```

After that you will need to choose one of builtin I18n middleware or write your own.

Builtin middlewares:

#### Simplel18nMiddleware

Simple I18n middleware.

Chooses language code from the User object received in event

```
__init__(i18n: i18n, i18n_key: str \mid None = 'i18n', middleware_key: str = 'i18n_middleware') \rightarrow None Create an instance of middleware
```

#### **Parameters**

- **i18n** instance of I18n
- i18n\_key context key for I18n instance
- middleware\_key context key for this middleware

#### Constl18nMiddleware

Const middleware chooses statically defined locale

```
\__init\_(locale: str, i18n: I18n, i18n_key: str | None = 'i18n', middleware_key: str = 'i18n_middleware') \rightarrow None
```

Create an instance of middleware

#### **Parameters**

- **i18n** instance of I18n
- i18n\_key context key for I18n instance
- middleware\_key context key for this middleware

#### FSMI18nMiddleware

```
class aiogram.utils.i18n.middleware.FSMI18nMiddleware(i18n: i18n, key: str = 'locale', i18n_key: str = None = 'i18n', middleware_key: str = 'i18n_middleware')
```

This middleware stores locale in the FSM storage

```
__init__(i18n: i18n, key: str = 'locale', i18n_key: str | None = 'i18n', middleware_key: str = 'i18n_middleware') \rightarrow None
```

Create an instance of middleware

#### **Parameters**

- **i18n** instance of I18n
- **i18n\_key** context key for I18n instance
- middleware\_key context key for this middleware

**async** set\_locale(state: FSMContext, locale: str)  $\rightarrow$  None

Write new locale to the storage

#### **Parameters**

- state instance of FSMContext
- locale new locale

#### I18nMiddleware

or define you own based on abstract I18nMiddleware middleware:

Abstract I18n middleware.

```
__init__(i18n: i18n, i18n_key: str \mid None = 'i18n', middleware_key: str = 'i18n_middleware') \rightarrow None Create an instance of middleware
```

#### **Parameters**

- **i18n** instance of I18n
- i18n\_key context key for I18n instance
- middleware\_key context key for this middleware

**abstract async get\_locale**(event: TelegramObject, data: Dict[str, Any])  $\rightarrow$  str

Detect current user locale based on event and context.

#### This method must be defined in child classes

#### **Parameters**

- event
- data

#### **Returns**

**setup**(*router*: Router, *exclude*:  $Set[str] \mid None = None$ )  $\rightarrow BaseMiddleware$  Register middleware for all events in the Router

#### 5

**Parameters** 

- router
- exclude

#### Returns

#### **Deal with Babel**

#### Step 1 Extract messages

```
pybabel extract --input-dirs=. -o locales/messages.pot
```

Here is --input-dirs=. - path to code and the locales/messages.pot is template where messages will be extracted and *messages* is translation domain.

#### Working with plural forms

Extracting with Pybabel all strings options:

- -k \_:1,1t -k \_:1,2 for both singular and plural
- -k \_\_ for lazy strings

```
pybabel extract -k _:1,1t -k _:1,2 -k __ --input-dirs=. -o locales/messages.pot
```

# 1 Note

Some useful options:

- Add comments for translators, you can use another tag if you want (TR) --add-comments=NOTE
- Contact email for bugreport --msgid-bugs-address=EMAIL
- Disable comments with string location in code --no-location
- Copyrights --copyright-holder=AUTHOR
- Set project name --project=MySuperBot
- Set version --version=2.2

## Step 2: Init language

```
pybabel init -i locales/messages.pot -d locales -D messages -l en
```

- -i locales/messages.pot pre-generated template
- -d locales translations directory
- -D messages translations domain
- -1 en language. Can be changed to any other valid language code (For example -1 uk for ukrainian language)

# Step 3: Translate texts

To open .po file you can use basic text editor or any PO editor, e.g. Poedit

Just open the file named locales/{language}/LC\_MESSAGES/messages.po and write translations

#### Step 4: Compile translations

```
pybabel compile -d locales -D messages
```

#### Step 5: Updating messages

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

- Step 5.1: regenerate pot file: command from step 1
- Step 5.2: update po files

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

- Step 5.3: update your translations: location and tools you know from step 3
- Step 5.4: compile mo files: command from step 4

#### 2.5.3 Chat action sender

#### Sender

class aiogram.utils.chat\_action.ChatActionSender(\*, bot: Bot, chat\_id:  $str \mid int, message\_thread\_id: int \mid None = None, action: <math>str = 'typing', interval: float = 5.0, initial\_sleep: float = 0.0$ )

This utility helps to automatically send chat action until long actions is done to take acknowledge bot users the bot is doing something and not crashed.

Provides simply to use context manager.

Technically sender start background task with infinity loop which works until action will be finished and sends the chat action every 5 seconds.

**\_\_init\_\_**(\*, bot: Bot, chat\_id:  $str \mid int$ ,  $message\_thread\_id$ :  $int \mid None = None$ , action: str = 'typing', interval: float = 5.0, initial sleep: float = 0.0)  $\rightarrow$  None

#### **Parameters**

- **bot** instance of the bot
- chat\_id target chat id
- message\_thread\_id unique identifier for the target message thread; supergroups only
- **action** chat action type
- **interval** interval between iterations
- initial\_sleep sleep before first sending of the action

**classmethod choose\_sticker**( $chat\_id$ :  $int \mid str$ , bot: Bot,  $message\_thread\_id$ :  $int \mid None = None$ , interval: float = 5.0,  $initial\_sleep$ : float = 0.0)  $\rightarrow ChatActionSender$ 

Create instance of the sender with choose\_sticker action

**classmethod find\_location**( $chat\_id$ :  $int \mid str, bot$ :  $Bot, message\_thread\_id$ :  $int \mid None = None, interval$ :  $float = 5.0, initial\_sleep$ : float = 0.0)  $\rightarrow ChatActionSender$ 

Create instance of the sender with find location action

**classmethod record\_video**(chat\_id: int | str, bot: Bot, message\_thread\_id: int | None = None, interval: float = 5.0, initial\_sleep: float = 0.0)  $\rightarrow$  ChatActionSender

Create instance of the sender with record\_video action

**classmethod record\_video\_note**( $chat_id: int \mid str, bot: Bot, message_thread_id: int \mid None = None, interval: float = 5.0, initial_sleep: float = 0.0) <math>\rightarrow ChatActionSender$ 

Create instance of the sender with record\_video\_note action

**classmethod record\_voice**(chat\_id: int | str, bot: Bot, message\_thread\_id: int | None = None, interval: float = 5.0, initial\_sleep: float = 0.0)  $\rightarrow$  ChatActionSender

Create instance of the sender with record voice action

**classmethod typing**( $chat\_id$ :  $int \mid str$ , bot: Bot,  $message\_thread\_id$ :  $int \mid None = None$ , interval: float = 5.0,  $initial\_sleep$ : float = 0.0)  $\rightarrow ChatActionSender$ 

Create instance of the sender with typing action

classmethod upload\_document(chat\_id: int | str, bot: Bot, message\_thread\_id: int | None = None, interval: float = 5.0, initial\_sleep: float = 0.0)  $\rightarrow$  ChatActionSender

Create instance of the sender with *upload\_document* action

```
classmethod upload_photo(chat_id: int | str, bot: Bot, message_thread_id: int | None = None, interval: float = 5.0, initial_sleep: float = 0.0) \rightarrow ChatActionSender
```

Create instance of the sender with upload\_photo action

classmethod upload\_video(chat\_id: int | str, bot: Bot, message\_thread\_id: int | None = None, interval: float = 5.0, initial\_sleep: float = 0.0)  $\rightarrow$  ChatActionSender

Create instance of the sender with upload\_video action

classmethod upload\_video\_note( $chat\_id$ :  $int \mid str, bot$ :  $Bot, message\_thread\_id$ :  $int \mid None = None,$  interval:  $float = 5.0, initial\_sleep$ : float = 0.0)  $\rightarrow ChatActionSender$ 

Create instance of the sender with upload\_video\_note action

**classmethod upload\_voice**( $chat\_id$ :  $int \mid str$ , bot: Bot,  $message\_thread\_id$ :  $int \mid None = None$ , interval: float = 5.0,  $initial\_sleep$ : float = 0.0)  $\rightarrow ChatActionSender$ 

Create instance of the sender with *upload\_voice* action

## **Usage**

```
async with ChatActionSender.typing(bot=bot, chat_id=message.chat.id):
    # Do something...
# Perform some long calculations
await message.answer(result)
```

#### **Middleware**

class aiogram.utils.chat\_action.ChatActionMiddleware

Helps to automatically use chat action sender for all message handlers

#### **Usage**

Before us a should be registered for the message event

```
<router or dispatcher>.message.middleware(ChatActionMiddleware())
```

After this action all handlers which works longer than *initial\_sleep* will produce the 'typing' chat action.

Also sender can be customized via flags feature for particular handler.

Change only action type:

```
@router.message(...)
@flags.chat_action("sticker")
async def my_handler(message: Message): ...
```

Change sender configuration:

```
@router.message(...)
@flags.chat_action(initial_sleep=2, action="upload_document", interval=3)
async def my_handler(message: Message): ...
```

## 2.5.4 WebApp

Telegram Bot API 6.0 announces a revolution in the development of chatbots using WebApp feature.

You can read more details on it in the official blog and documentation.

aiogram implements simple utils to remove headache with the data validation from Telegram WebApp on the backend side.

#### **Usage**

For example from frontend you will pass application/x-www-form-urlencoded POST request with \_auth field in body and wants to return User info inside response as application/json

```
from aiogram.utils.web_app import safe_parse_webapp_init_data
from aiohttp.web_request import Request
from aiohttp.web_response import json_response

async def check_data_handler(request: Request):
   bot: Bot = request.app["bot"]

   data = await request.post() # application/x-www-form-urlencoded
   try:
      data = safe_parse_webapp_init_data(token=bot.token, init_data=data["_auth"])
   except ValueError:
      return json_response({"ok": False, "err": "Unauthorized"}, status=401)
      return json_response({"ok": True, "data": data.user.dict()})
```

#### **Functions**

```
aiogram.utils.web_app.check_webapp_signature(token: str, init_data: str) → bool
```

Check incoming WebApp init data signature

Source: https://core.telegram.org/bots/webapps#validating-data-received-via-the-web-app

#### Parameters

- token bot Token
- init\_data data from frontend to be validated

#### **Returns**

```
aiogram.utils.web_app.parse_webapp_init_data(init_data: str, *, loads: ~typing.Callable[[...], \simtyping.Any] = <function loads>) \rightarrow WebAppInitData
```

Parse WebApp init data and return it as WebAppInitData object

This method doesn't make any security check, so you shall not trust to this data, use safe\_parse\_webapp\_init\_data instead.

#### **Parameters**

- init\_data data from frontend to be parsed
- loads

#### Returns

Validate raw WebApp init data and return it as WebAppInitData object

Raise ValueError when data is invalid

#### **Parameters**

- token bot token
- init\_data data from frontend to be parsed and validated
- loads

#### **Returns**

## **Types**

```
class aiogram.utils.web_app.WebAppInitData(**extra_data: Any)
```

This object contains data that is transferred to the Web App when it is opened. It is empty if the Web App was launched from a keyboard button.

Source: https://core.telegram.org/bots/webapps#webappinitdata

```
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
```

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

```
model_config: ClassVar[ConfigDict] = {'arbitrary_types_allowed': True,
  'defer_build': True, 'extra': 'allow', 'frozen': True, 'populate_by_name': True,
  'use_enum_values': True, 'validate_assignment': True}
```

Configuration for the model, should be a dictionary conforming to [Config-Dict][pydantic.config.ConfigDict].

```
model_fields: ClassVar[Dict[str, FieldInfo]] = {'auth_date':
FieldInfo(annotation=datetime, required=True), 'can_send_after':
FieldInfo(annotation=Union[int, NoneType], required=False, default=None), 'chat':
FieldInfo(annotation=Union[WebAppChat, NoneType], required=False, default=None),
'chat_instance': FieldInfo(annotation=Union[str, NoneType], required=False,
default=None), 'chat_type': FieldInfo(annotation=Union[str, NoneType],
required=False, default=None), 'hash': FieldInfo(annotation=str, required=True),
'query_id': FieldInfo(annotation=Union[str, NoneType], required=False,
default=None), 'receiver': FieldInfo(annotation=Union[WebAppUser, NoneType],
required=False, default=None), 'start_param': FieldInfo(annotation=Union[str,
NoneType], required=False, default=None), 'user':
FieldInfo(annotation=Union[WebAppUser, NoneType], required=False, default=None)}
```

Metadata about the fields defined on the model, mapping of field names to [Field-Info][pydantic.fields.FieldInfo] objects.

This replaces *Model*. *fields* from Pydantic V1.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

```
query_id: str | None
```

A unique identifier for the Web App session, required for sending messages via the answerWebAppQuery method.

#### user: WebAppUser | None

An object containing data about the current user.

#### receiver: WebAppUser | None

An object containing data about the chat partner of the current user in the chat where the bot was launched via the attachment menu. Returned only for Web Apps launched via the attachment menu.

#### chat: WebAppChat | None

An object containing data about the chat where the bot was launched via the attachment menu. Returned for supergroups, channels, and group chats – only for Web Apps launched via the attachment menu.

#### chat\_type: str | None

Type of the chat from which the Web App was opened. Can be either "sender" for a private chat with the user opening the link, "private", "group", "supergroup", or "channel". Returned only for Web Apps launched from direct links.

#### chat\_instance: str | None

Global identifier, uniquely corresponding to the chat from which the Web App was opened. Returned only for Web Apps launched from a direct link.

#### start\_param: str | None

The value of the startattach parameter, passed via link. Only returned for Web Apps when launched from the attachment menu via link. The value of the start\_param parameter will also be passed in the GET-parameter tgWebAppStartParam, so the Web App can load the correct interface right away.

#### can\_send\_after: int | None

Time in seconds, after which a message can be sent via the answerWebAppQuery method.

#### auth\_date: datetime

Unix time when the form was opened.

#### hash: str

A hash of all passed parameters, which the bot server can use to check their validity.

#### class aiogram.utils.web\_app.WebAppUser(\*\*extra\_data: Any)

This object contains the data of the Web App user.

Source: https://core.telegram.org/bots/webapps#webappuser

#### id: int

A unique identifier for the user or bot. This number may have more than 32 significant bits and some programming languages may have difficulty/silent defects in interpreting it. It has at most 52 significant bits, so a 64-bit integer or a double-precision float type is safe for storing this identifier.

#### is\_bot: bool | None

True, if this user is a bot. Returns in the receiver field only.

#### first\_name: str

First name of the user or bot.

#### last\_name: str | None

Last name of the user or bot.

#### username: str | None

Username of the user or bot.

#### language\_code: str | None

IETF language tag of the user's language. Returns in user field only.

# is\_premium: bool | None True, if this user is a Telegram Premium user. added\_to\_attachment\_menu: bool | None True, if this user added the bot to the attachment menu. allows\_write\_to\_pm: bool | None True, if this user allowed the bot to message them.

model\_computed\_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

```
model_config: ClassVar[ConfigDict] = {'arbitrary_types_allowed': True,
'defer_build': True, 'extra': 'allow', 'frozen': True, 'populate_by_name': True,
'use_enum_values': True, 'validate_assignment': True}
```

Configuration for the model, should be a dictionary conforming to [Config-Dict][pydantic.config.ConfigDict].

```
model_fields: ClassVar[Dict[str, FieldInfo]] = {'added_to_attachment_menu':
    FieldInfo(annotation=Union[bool, NoneType], required=False, default=None),
    'allows_write_to_pm': FieldInfo(annotation=Union[bool, NoneType], required=False,
    default=None), 'first_name': FieldInfo(annotation=str, required=True), 'id':
    FieldInfo(annotation=int, required=True), 'is_bot':
    FieldInfo(annotation=Union[bool, NoneType], required=False, default=None),
    'is_premium': FieldInfo(annotation=Union[bool, NoneType], required=False,
    default=None), 'language_code': FieldInfo(annotation=Union[str, NoneType],
    required=False, default=None), 'last_name': FieldInfo(annotation=Union[str,
    NoneType], required=False, default=None), 'photo_url':
    FieldInfo(annotation=Union[str, NoneType], required=False, default=None),
    'username': FieldInfo(annotation=Union[str, NoneType], required=False,
    default=None)}
```

Metadata about the fields defined on the model, mapping of field names to [Field-Info][pydantic.fields.FieldInfo] objects.

This replaces *Model*.\_\_fields\_\_ from Pydantic V1.

```
model_post_init(context: Any, /) \rightarrow None
```

We need to both initialize private attributes and call the user-defined model post init method.

```
photo_url: str | None
```

URL of the user's profile photo. The photo can be in .jpeg or .svg formats. Only returned for Web Apps launched from the attachment menu.

```
class aiogram.utils.web_app.WebAppChat(**extra_data: Any)
```

This object represents a chat.

Source: https://core.telegram.org/bots/webapps#webappchat

#### id: int

Unique identifier for this chat. This number may have more than 32 significant bits and some programming languages may have difficulty/silent defects in interpreting it. But it has at most 52 significant bits, so a signed 64-bit integer or double-precision float type are safe for storing this identifier.

#### type: str

Type of chat, can be either "group", "supergroup" or "channel"

```
title: str
    Title of the chat
username: str | None
    Username of the chat
photo_url: str | None
    URL of the chat's photo. The photo can be in .jpeg or .svg formats. Only returned for Web Apps launched
    from the attachment menu.
model_computed_fields: ClassVar[Dict[str, ComputedFieldInfo]] = {}
    A dictionary of computed field names and their corresponding ComputedFieldInfo objects.
model_config: ClassVar[ConfigDict] = {'arbitrary_types_allowed': True,
'defer_build': True, 'extra': 'allow', 'frozen': True, 'populate_by_name':
'use_enum_values': True, 'validate_assignment': True}
    Configuration
                 for the
                             model,
                                      should
                                              be
                                                        dictionary
                                                                   conforming
                                                                                    [Config-
    Dict][pydantic.config.ConfigDict].
model_fields: ClassVar[Dict[str, FieldInfo]] = {'id': FieldInfo(annotation=int,
required=True), 'photo_url': FieldInfo(annotation=Union[str, NoneType],
required=False, default=None), 'title': FieldInfo(annotation=str, required=True),
'type': FieldInfo(annotation=str, required=True), 'username':
FieldInfo(annotation=Union[str, NoneType], required=False, default=None)}
    Metadata about the fields defined on the model, mapping of field names to [Field-
    Info][pydantic.fields.FieldInfo] objects.
    This replaces Model.__fields__ from Pydantic V1.
```

We need to both initialize private attributes and call the user-defined model\_post\_init method.

 $model_post_init(context: Any, /) \rightarrow None$ 

#### 2.5.5 Callback answer

Helper for callback query handlers, can be useful in bots with a lot of callback handlers to automatically take answer to all requests.

#### Simple usage

For use, it is enough to register the inner middleware aiogram.utils.callback\_answer. CallbackAnswerMiddleware in dispatcher or specific router:

```
dispatcher.callback_query.middleware(CallbackAnswerMiddleware())
```

After that all handled callback queries will be answered automatically after processing the handler.

#### Advanced usage

In some cases you need to have some non-standard response parameters, this can be done in several ways:

#### Global defaults

Change default parameters while initializing middleware, for example change answer to pre mode and text "OK":

```
dispatcher.callback_query.middleware(CallbackAnswerMiddleware(pre=True, text="OK"))
```

Look at aiogram.utils.callback\_answer.CallbackAnswerMiddleware to get all available parameters

#### Handler specific

By using flags you can change the behavior for specific handler

```
@router.callback_query(<filters>)
@flags.callback_answer(text="Thanks", cache_time=30)
async def my_handler(query: CallbackQuery):
...
```

Flag arguments is the same as in aiogram.utils.callback\_answer.CallbackAnswerMiddleware with additional one disabled to disable answer.

#### A special case

It is not always correct to answer the same in every case, so there is an option to change the answer inside the handler. You can get an instance of <code>aiogram.utils.callback\_answer.CallbackAnswer</code> object inside handler and change whatever you want.

# **Danger**

Note that is impossible to change callback answer attributes when you use pre=True mode.

#### Combine that all at once

For example you want to answer in most of cases before handler with text "" but at some cases need to answer after the handler with custom text, so you can do it:

```
dispatcher.callback_query.middleware(CallbackAnswerMiddleware(pre=True, text=""))
@router.callback_query(<filters>)
@flags.callback_answer(pre=False, cache_time=30)
async def my_handler(query: CallbackQuery):
    ...
    if <everything is ok>:
        callback_answer.text = "All is ok"
```

#### **Description of objects**

Bases: BaseMiddleware

```
__init__(pre: bool = False, text: str | None = None, show_alert: bool | None = None, url: str | None = None, cache_time: int | None = None) \rightarrow None
```

Inner middleware for callback query handlers, can be useful in bots with a lot of callback handlers to automatically take answer to all requests

#### **Parameters**

- pre send answer before execute handler
- text answer with text
- show\_alert show alert
- url game url
- **cache\_time** cache answer for some time

class aiogram.utils.callback\_answer.CallbackAnswer(answered: bool, disabled: bool = False, text:  $str \mid None = None$ ,  $show\_alert$ :  $bool \mid None = None$ , url:  $str \mid None = None$ ,  $cache\_time$ :  $int \mid None = None$ )

Bases: object

```
__init__(answered: bool, disabled: bool = False, text: str | None = None, show_alert: bool | None = None, url: str | None = None, cache_time: int | None = None) \rightarrow None
```

Callback answer configuration

#### **Parameters**

- answered this request is already answered by middleware
- **disabled** answer will not be performed
- text answer with text
- **show\_alert** show alert

```
    url – game url
    cache_time – cache answer for some time
    disable() → None
        Deactivate answering for this handler
    property disabled: bool
        Indicates that automatic answer is disabled in this handler
    property answered: bool
        Indicates that request is already answered by middleware
    property text: str | None
        Response text:return:
    property show_alert: bool | None
        Whether to display an alert
    property url: str | None
        Game url
```

# 2.5.6 Formatting

Make your message formatting flexible and simple

property cache\_time: int | None

Response cache time

This instrument works on top of Message entities instead of using HTML or Markdown markups, you can easily construct your message and sent it to the Telegram without the need to remember tag parity (opening and closing) or escaping user input.

#### **Usage**

#### **Basic scenario**

Construct your message and send it to the Telegram.

```
content = Text("Hello, ", Bold(message.from_user.full_name), "!")
await message.answer(**content.as_kwargs())
```

Is the same as the next example, but without usage markup

```
await message.answer(
    text=f"Hello, <b>{html.quote(message.from_user.full_name)}!",
    parse_mode=ParseMode.HTML
)
```

Literally when you execute as\_kwargs method the Text object is converted into text Hello, Alex! with entities list [MessageEntity(type='bold', offset=7, length=4)] and passed into dict which can be used as \*\*kwargs in API call.

The complete list of elements is listed on this page below.

#### **Advanced scenario**

On top of base elements can be implemented content rendering structures, so, out of the box aiogram has a few already implemented functions that helps you to format your messages:

```
aiogram.utils.formatting.as_line(*items: Any, end: str = \n', sep: str = \n') \rightarrow Text
```

Wrap multiple nodes into line with \n at the end of line.

#### **Parameters**

- items Text or Any
- end ending of the line, by default is \n
- sep separator between items, by default is empty string

#### Returns

Text

aiogram.utils.formatting.as\_list(\*items: Any, sep:  $str = \n'$ )  $\rightarrow Text$ 

Wrap each element to separated lines

#### **Parameters**

- items
- sep

#### Returns

 $\verb|aiogram.utils.formatting.as_marked_list(*items: Any, marker: str = '-')| \rightarrow \textit{Text}|$ 

Wrap elements as marked list

#### **Parameters**

- items
- marker line marker, by default is '- '

#### Returns

Text

aiogram.utils.formatting.as\_numbered\_list(\*items: Any, start: int = 1, fmt:  $str = '\{\}$ . ')  $\rightarrow Text$  Wrap elements as numbered list

#### **Parameters**

- items
- start initial number, by default 1
- **fmt** number format, by default '{}. '

#### Returns

Text

aiogram.utils.formatting.as\_section(title: Any, \*body: Any)  $\rightarrow$  Text

Wrap elements as simple section, section has title and body

#### **Parameters**

- title
- body

#### Returns

Text

aiogram.utils.formatting.as\_marked\_section( $title: Any, *body: Any, marker: str = '- ') \rightarrow Text$ Wrap elements as section with marked list

#### **Parameters**

- title
- · body
- marker

#### Returns

aiogram.utils.formatting.as\_numbered\_section(title: Any, \*body: Any, start: int = 1, fmt:  $str = '\{\}.$ ')  $\rightarrow Text$ 

Wrap elements as section with numbered list

#### **Parameters**

- title
- · body
- start
- fmt

#### Returns

aiogram.utils.formatting.as\_key\_value(key: Any, value: Any)  $\rightarrow Text$ Wrap elements pair as key-value line. (<b>{key}:</b> {value})

#### **Parameters**

- key
- value

#### Returns

Text

and lets complete them all:

```
content = as_list(
    as_marked_section(
        Bold("Success:"),
        "Test 1",
        "Test 3",
        "Test 4"
        marker=" ",
    ),
    as_marked_section(
        Bold("Failed:"),
        "Test 2",
        marker=" ",
    ),
    as_marked_section(
        Bold("Summary:"),
        as_key_value("Total", 4),
```

(continues on next page)

(continued from previous page)

```
as_key_value("Success", 3),
    as_key_value("Failed", 1),
    marker=" ",
    ),
    HashTag("#test"),
    sep="\n\n",
)
```

Will be rendered into:

#### **Success:**

Test 1

Test 3

Test 4

#### Failed:

Test 2

# **Summary:**

Total: 4

Success: 3

Failed: 1

#test

Or as HTML:

#### **Available methods**

```
class aiogram.utils.formatting.Text(*body: Any, **params: Any)
                   Bases: Iterable[Any]
                   Simple text element
                   __init__(*body: Any, **params: Any) \rightarrow None
                   render(*, \_offset: int = 0, \_sort: bool = True, \_collect\_entities: bool = True) \rightarrow Tuple[str, \_offset: int = 0, \_sort: bool = True, \_collect\_entities: bool = True) \rightarrow Tuple[str, \_offset: int = 0, \_sort: bool = True, \_collect\_entities: bool = True) \rightarrow Tuple[str, \_offset: int = 0, \_sort: bool = True, \_collect\_entities: bool = True) \rightarrow Tuple[str, \_offset: int = 0, \_sort: bool = True, \_collect\_entities: bool = True) \rightarrow Tuple[str, \_offset: int = 0, \_sort: bool = True, \_collect\_entities: bool = True, \_collect\_enti
                                                 List[MessageEntity]]
                                     Render elements tree as text with entities list
                                                    Returns
                   as_kwargs(*, text_key: str = 'text', entities_key: str = 'entities', replace_parse_mode: bool = True,
                                                             parse\_mode\_key: str = 'parse\_mode') \rightarrow Dict[str, Any]
                                     Render elements tree as keyword arguments for usage in the API call, for example:
                                     entities = Text(...)
                                     await message.answer(**entities.as_kwargs())
                                                    Parameters
                                                                   text_key
                                                                   entities_key
                                                                   • replace_parse_mode
                                                                   • parse_mode_key
                                                    Returns
                   as\_html() \rightarrow str
                                     Render elements tree as HTML markup
                   as_markdown() \rightarrow str
                                     Render elements tree as MarkdownV2 markup
Available elements
                   Bases: Iterable[Any]
```

```
class aiogram.utils.formatting.Text(*body: Any, **params: Any)
     Simple text element
class aiogram.utils.formatting.HashTag(*body: Any, **params: Any)
     Bases: Text
     Hashtag element.
```

#### Warning

The value should always start with '#' symbol

Will be wrapped into aiogram.types.message\_entity.MessageEntity with type aiogram.enums. message\_entity\_type.MessageEntityType.HASHTAG

class aiogram.utils.formatting.CashTag(\*body: Any, \*\*params: Any)

Bases: Text

Cashtag element.

#### Warning

The value should always start with '\$' symbol

Will be wrapped into aiogram.types.message\_entity.MessageEntity with type aiogram.enums. message\_entity\_type.MessageEntityType.CASHTAG

**class** aiogram.utils.formatting.**BotCommand**(\*body: Any, \*\*params: Any)

Bases: Text

Bot command element.

#### Warning

The value should always start with '/' symbol

Will be wrapped into aiogram.types.message\_entity.MessageEntity with type aiogram.enums. message\_entity\_type.MessageEntityType.BOT\_COMMAND

**class** aiogram.utils.formatting.**Url**(\*body: Any, \*\*params: Any)

Bases: Text

Url element.

Will be wrapped into aiogram.types.message\_entity.MessageEntity with type aiogram.enums. message\_entity\_type.MessageEntityType.URL

class aiogram.utils.formatting.Email(\*body: Any, \*\*params: Any)

Bases: Text

Email element.

Will be wrapped into aiogram.types.message\_entity.MessageEntity with type aiogram.enums. message\_entity\_type.MessageEntityType.EMAIL

class aiogram.utils.formatting.PhoneNumber(\*body: Any, \*\*params: Any)

Bases: Text

Phone number element.

Will be wrapped into aiogram.types.message\_entity.MessageEntity with type aiogram.enums. message\_entity\_type.MessageEntityType.PHONE\_NUMBER

class aiogram.utils.formatting.Bold(\*body: Any, \*\*params: Any)

Bases: Text

Bold element.

Will be wrapped into aiogram.types.message\_entity.MessageEntity with type aiogram.enums. message\_entity\_type.MessageEntityType.BOLD

```
class aiogram.utils.formatting.Italic(*body: Any, **params: Any)
     Bases: Text
     Italic element.
     Will be wrapped into aiogram.types.message_entity.MessageEntity with type aiogram.enums.
     message_entity_type.MessageEntityType.ITALIC
class aiogram.utils.formatting.Underline(*body: Any, **params: Any)
     Bases: Text
     Underline element.
     Will be wrapped into aiogram.types.message_entity.MessageEntity with type aiogram.enums.
     message_entity_type.MessageEntityType.UNDERLINE
class aiogram.utils.formatting.Strikethrough(*body: Any, **params: Any)
     Bases: Text
     Strikethrough element.
     Will be wrapped into aiogram.types.message_entity.MessageEntity with type aiogram.enums.
     message_entity_type.MessageEntityType.STRIKETHROUGH
class aiogram.utils.formatting.Spoiler(*body: Any, **params: Any)
     Bases: Text
     Spoiler element.
     Will be wrapped into aiogram.types.message_entity.MessageEntity with type aiogram.enums.
     message_entity_type.MessageEntityType.SPOILER
class aiogram.utils.formatting.Code(*body: Any, **params: Any)
     Bases: Text
     Code element.
     Will be wrapped into aiogram.types.message_entity.MessageEntity with type aiogram.enums.
     message_entity_type.MessageEntityType.CODE
class aiogram.utils.formatting.Pre(*body: Any, language: str | None = None, **params: Any)
     Bases: Text
     Pre element.
     Will be wrapped into aiogram.types.message_entity.MessageEntity with type aiogram.enums.
     message_entity_type.MessageEntityType.PRE
class aiogram.utils.formatting.TextLink(*body: Any, url: str, **params: Any)
     Bases: Text
     Text link element.
     Will be wrapped into aiogram.types.message_entity.MessageEntity with type aiogram.enums.
     message_entity_type.MessageEntityType.TEXT_LINK
class aiogram.utils.formatting.TextMention(*body: Any, user: User, **params: Any)
     Bases: Text
     Text mention element.
     Will be wrapped into aiogram.types.message_entity.MessageEntity with type aiogram.enums.
     message_entity_type.MessageEntityType.TEXT_MENTION
```

**class** aiogram.utils.formatting.**CustomEmoji**(\*body: Any, custom\_emoji\_id: str, \*\*params: Any)

Bases: Text

Custom emoji element.

Will be wrapped into aiogram.types.message\_entity.MessageEntity with type aiogram.enums. message\_entity\_type.MessageEntityType.CUSTOM\_EMOJI

class aiogram.utils.formatting.BlockQuote(\*body: Any, \*\*params: Any)

Bases: Text

Block quote element.

Will be wrapped into aiogram.types.message\_entity.MessageEntity with type aiogram.enums. message\_entity\_type.MessageEntityType.BLOCKQUOTE

**class** aiogram.utils.formatting.**ExpandableBlockQuote**(\*body: Any, \*\*params: Any)

Bases: Text

Expandable block quote element.

Will be wrapped into aiogram.types.message\_entity.MessageEntity with type aiogram.enums. message\_entity\_type.MessageEntityType.EXPANDABLE\_BLOCKQUOTE

# 2.5.7 Media group builder

This module provides a builder for media groups, it can be used to build media groups for aiogram.types. input\_media\_photo.InputMediaPhoto, aiogram.types.input\_media\_video.InputMediaVideo, aiogram.types.input\_media\_document.InputMediaDocument and aiogram.types.input\_media\_audio. InputMediaAudio.

#### Warning

aiogram.types.input\_media\_animation.InputMediaAnimation is not supported yet in the Bot API to send as media group.

# **Usage**

```
media_group = MediaGroupBuilder(caption="Media group caption")
# Add photo
media_group.add_photo(media="https://picsum.photos/200/300")
# Dynamically add photo with known type without using separate method
media_group.add(type="photo", media="https://picsum.photos/200/300")
# ... or video
media_group.add(type="video", media=FSInputFile("media/video.mp4"))
```

To send media group use aiogram.methods.send\_media\_group.SendMediaGroup() method, but when you use aiogram.utils.media\_group.MediaGroupBuilder you should pass media argument as media\_group. build().

If you specify caption in aiogram.utils.media\_group.MediaGroupBuilder it will be used as caption for first media in group.

await bot.send\_media\_group(chat\_id=chat\_id, media=media\_group.build())

#### References

- add(\*, type: Literal[InputMediaType.AUDIO], media: str | InputFile, caption: str | None = None, parse\_mode: str | None = UNSET\_PARSE\_MODE, caption\_entities: List[MessageEntity] | None = None, duration: int | None = None, performer: str | None = None, title: str | None = None, \*\*kwargs: Any) → None
- add(\*, type: Literal[InputMediaType.PHOTO], media: str | InputFile, caption: str | None = None,
   parse\_mode: str | None = UNSET\_PARSE\_MODE, caption\_entities: List[MessageEntity] | None =
   None, has\_spoiler: bool | None = None, \*\*kwargs: Any) → None
- add(\*, type: Literal[InputMediaType.VIDEO], media: str | InputFile, thumbnail: InputFile | str | None = None, caption: str | None = None, parse\_mode: str | None = UNSET\_PARSE\_MODE, caption\_entities: List[MessageEntity] | None = None, width: int | None = None, height: int | None = None, duration: int | None = None, supports\_streaming: bool | None = None, has\_spoiler: bool | None = None, \*\*kwargs: Any) → None
- $\begin{array}{l} \textbf{add}(*, type: Literal[InputMediaType.DOCUMENT], media: str \mid InputFile, thumbnail: InputFile \mid str \mid None \\ = None, caption: str \mid None = None, parse\_mode: str \mid None = UNSET\_PARSE\_MODE, caption\_entities: \\ List[MessageEntity] \mid None = None, disable\_content\_type\_detection: bool \mid None = None, **kwargs: \\ Any) \rightarrow \text{None} \\ \end{array}$

Add a media object to the media group.

#### **Parameters**

**kwargs** – Keyword arguments for the media object. The available keyword arguments depend on the media type.

#### Returns

None

add\_audio(media: str | ~aiogram.types.input\_file.InputFile, thumbnail: ~aiogram.types.input\_file.InputFile | None = None, caption: str | None = None, parse\_mode: str | None = <Default('parse\_mode')>, caption\_entities: ~typing.List[~aiogram.types.message\_entity.MessageEntity] | None = None, duration: int | None = None, performer: str | None = None, title: str | None = None, \*\*kwargs: ~typing.Any) → None

Add an audio file to the media group.

#### **Parameters**

• **media** – File to send. Pass a file\_id to send a file that exists on the Telegram servers (recommended), pass an HTTP URL for Telegram to get a file from the Internet, or pass 'attach://<file\_attach\_name>' to upload a new one using multipart/form-data under <file\_attach\_name> name.

More information on Sending Files »

• **thumbnail** – *Optional*. Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320.

- caption Optional. Caption of the audio to be sent, 0-1024 characters after entities parsing
- parse\_mode *Optional*. Mode for parsing entities in the audio caption. See formatting options for more details.
- **caption\_entities** *Optional*. List of special entities that appear in the caption, which can be specified instead of *parse\_mode*
- duration Optional. Duration of the audio in seconds
- **performer** *Optional*. Performer of the audio
- title Optional. Title of the audio

#### Returns

None

add\_document(media: str | ~aiogram.types.input\_file.InputFile, thumbnail:

```
~aiogram.types.input_file.InputFile | None = None, caption: str | None = None, parse_mode: str | None = <Default('parse_mode')>, caption_entities: 
~typing.List[~aiogram.types.message entity.MessageEntity] | None = None,
```

 $disable\_content\_type\_detection: bool \mid None = None, **kwargs: ~typing.Any) \rightarrow None$ 

Add a document to the media group.

#### **Parameters**

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

#### Returns

None

```
 \begin{tabular}{ll} \textbf{add\_photo}(\textit{media: str} \mid \sim & aiogram.types.input\_file.InputFile, caption: str \mid None = None, parse\_mode: str \mid None = <Default('parse\_mode')>, caption\_entities: \\ \sim & typing.List[\sim & aiogram.types.message\_entity.MessageEntity] \mid None = None, has\_spoiler: bool \mid None = None, **kwargs: \sim & typing.Any) \rightarrow None \\ \end{tabular}
```

Add a photo to the media group.

#### **Parameters**

• **media** – File to send. Pass a file\_id to send a file that exists on the Telegram servers (recommended), pass an HTTP URL for Telegram to get a file from the Internet, or pass 'attach://<file\_attach\_name>' to upload a new one using multipart/form-data under <file\_attach\_name> name.

More information on Sending Files »

- **caption** *Optional*. Caption of the photo to be sent, 0-1024 characters after entities parsing
- parse\_mode *Optional*. Mode for parsing entities in the photo caption. See formatting options for more details.
- **caption\_entities** *Optional*. List of special entities that appear in the caption, which can be specified instead of *parse\_mode*
- has\_spoiler Optional. Pass True if the photo needs to be covered with a spoiler animation

#### Returns

None

add\_video(media: str | ~aiogram.types.input\_file.InputFile, thumbnail: ~aiogram.types.input\_file.InputFile | None = None, caption: str | None = None, parse\_mode: str | None = <Default('parse\_mode')>, caption\_entities: ~typing.List[~aiogram.types.message\_entity.MessageEntity] | None = None, width: int | None = None, height: int | None = None, duration: int | None = None, supports\_streaming: bool | None = None, has\_spoiler: bool | None = None, \*\*kwargs: ~typing.Any) → None

Add a video to the media group.

#### **Parameters**

- **media** File to send. Pass a file\_id to send a file that exists on the Telegram servers (recommended), pass an HTTP URL for Telegram to get a file from the Internet, or pass 'attach://<file\_attach\_name>' to upload a new one using multipart/form-data under <file\_attach\_name> name. *More information on Sending Files* »
- **thumbnail** *Optional*. Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file\_attach\_name>' if the thumbnail was uploaded using multipart/form-data under <file\_attach\_name>. *More information on Sending Files* »
- caption Optional. Caption of the video to be sent, 0-1024 characters after entities parsing
- parse\_mode *Optional*. Mode for parsing entities in the video caption. See formatting options for more details.
- **caption\_entities** *Optional*. List of special entities that appear in the caption, which can be specified instead of *parse\_mode*
- width Optional. Video width
- **height** *Optional*. Video height
- **duration** *Optional*. Video duration in seconds

- **supports\_streaming** *Optional*. Pass True if the uploaded video is suitable for streaming
- has\_spoiler Optional. Pass True if the video needs to be covered with a spoiler animation

#### Returns

None

build() → List[InputMediaAudio | InputMediaPhoto | InputMediaVideo | InputMediaDocument]

Builds a list of media objects for a media group.

Adds the caption to the first media object if it is present.

#### Returns

List of media objects.

# 2.5.8 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/features#deep-linking

We have added some utils to get deep links more handy.

#### **Examples**

#### Basic link example

```
from aiogram.utils.deep_linking import create_start_link
link = await create_start_link(bot, 'foo')
# result: 'https://t.me/MyBot?start=foo'
```

#### **Encoded link**

```
from aiogram.utils.deep_linking import create_start_link
link = await create_start_link(bot, 'foo', encode=True)
# result: 'https://t.me/MyBot?start=Zm9v'
```

#### Decode it back

```
from aiogram.utils.deep_linking import decode_payload
from aiogram.filters import CommandStart, CommandObject
from aiogram.types import Message

@router.message(CommandStart(deep_link=True))
async def handler(message: Message, command: CommandObject):
    args = command.args
    payload = decode_payload(args)
    await message.answer(f"Your payload: {payload}")
```

#### References

```
async aiogram.utils.deep_linking.create_start_link(bot: Bot, payload: str, encode: bool = False, encoder: Callable[[bytes], bytes] | None = None) \rightarrow str
```

Create 'start' deep link with your payload.

If you need to encode payload or pass special characters -

set encode as True

#### **Parameters**

- **bot** bot instance
- payload args passed with /start
- encode encode payload with base64url or custom encoder
- encoder custom encoder callable

#### Returns

link

```
aiogram.utils.deep_linking.decode_payload(payload: str, decoder: Callable[[bytes], bytes] | None = None) \rightarrow str
```

Decode URL-safe base64url payload with decoder.

# 2.5.9 Telegram object serialization

#### Serialization

To serialize Python object to Telegram object you can use pydantic serialization methods, for example:

```
message_data = { ... } # Some message data as dict
message = Message.model_validate(message_data)
```

If you want to bind serialized object to the Bot instance, you can use context:

```
message_data = { ... } # Some message data as dict
message = Message.model_validate(message_data, context={"bot": bot})
```

#### **Deserialization**

In cases when you need to deserialize Telegram object to Python object, you can use this module.

To convert Telegram object to Python object excluding files you can use aiogram.utils.serialization. deserialize\_telegram\_object\_to\_python() function.

aiogram.utils.serialization.deserialize\_telegram\_object\_to\_python(obj: Any, default:

DefaultBotProperties | None = None, include\_api\_method\_name: bool = True)  $\rightarrow$  Any

Deserialize telegram object to JSON compatible Python object excluding files.

#### **Parameters**

- **obj** The telegram object to be deserialized.
- **default** Default bot properties should be passed only if you want to use custom defaults.
- include\_api\_method\_name Whether to include the API method name in the result.

#### Returns

The deserialized telegram object.

To convert Telegram object to Python object including files you can use aiogram.utils.serialization. deserialize\_telegram\_object() function, which returns aiogram.utils.serialization. DeserializedTelegramObject object.

```
aiogram.utils.serialization.deserialize_telegram_object(obj: Any, default: DefaultBotProperties | None = None, include_api_method_name: bool = True) \rightarrow DeserializedTelegramObject
```

Deserialize Telegram Object to JSON compatible Python object.

#### **Parameters**

- **obj** The object to be deserialized.
- **default** Default bot properties should be passed only if you want to use custom defaults.
- include\_api\_method\_name Whether to include the API method name in the result.

#### Returns

The deserialized Telegram object.

class aiogram.utils.serialization.DeserializedTelegramObject(data: Any, files: Dict[str, InputFile])
 Represents a dumped Telegram object.

#### **Parameters**

- data (Any) The dumped data of the Telegram object.
- **files** (*Dict[str*, InputFile]) The dictionary containing the file names as keys and the corresponding *InputFile* objects as values.

# 2.6 Changelog

# 2.6.1 3.13.1 (2024-09-18)

#### Warning

Python 3.8 End of Life: Python 3.8 will reach its end of life (EOL) soon and will no longer be supported by aiogram in the next releases (1-2 months ETA).

Please upgrade to a newer version of Python to ensure compatibility and receive future updates.

#### Misc

- Increase max pydantic version support "<2.9" -> "<2.10" (only For Python >=3.9) #1576
- Bump aiofiles version upper bound to <24.2 #1577

# **Bugfixes**

• Fixed *Default* object annotation resolution using *pydantic* #1579

# 2.6.2 3.13.0 (2024-09-08)

#### **Features**

- Added updates about purchased paid media, represented by the class aiogram.types. paid\_media\_purchased.PaidMediaPurchased and the field purchased\_paid\_media in the class aiogram.types.update.Update.
  - Added the ability to specify a payload in aiogram.methods.send\_paid\_media.SendPaidMedia that is received back by the bot in aiogram.types.transaction\_partner\_user. TransactionPartnerUser and purchased\_paid\_media updates.
  - Added the field prize\_star\_count to the classes aiogram.types.giveaway\_created. GiveawayCreated, aiogram.types.giveaway.Giveaway, aiogram.types. giveaway\_winners.GiveawayWinners and aiogram.types.chat\_boost\_source\_giveaway. ChatBoostSourceGiveaway.
  - Added the field is\_star\_giveaway to the class aiogram.types.giveaway\_completed. GiveawayCompleted.

#### #1510

• Added missing method aliases such as .answer(), .reply(), and others to InaccessibleMessage. This change ensures consistency and improves usability by aligning the functionality of *InaccessibleMessage* with the *Message* type. #1574

2.6. Changelog 675

#### **Bugfixes**

• Fixed link preview options to use global defaults in various types and methods to use global defaults for link\_preview\_options. This change ensures consistency and enhances flexibility in handling link preview options across different components. #1543

## 2.6.3 3.12.0 (2024-08-16)

#### **Features**

- Added message\_thread\_id parameter to message.get\_url(). #1451
- Added getting user from chat\_boost with source ChatBoostSourcePremium in UserContextMiddleware for EventContext #1474
- Added full support of Bot API 7.8
  - Added the ability to send paid media to any chat.
  - Added the parameter business\_connection\_id to the method aiogram.methods. send\_paid\_media.SendPaidMedia, allowing bots to send paid media on behalf of a business account.
  - Added the field paid\_media to the class aiogram.types.transaction\_partner\_user.
     TransactionPartnerUser for transactions involving paid media.
  - Added the method aiogram.methods.create\_chat\_subscription\_invite\_link.
     CreateChatSubscriptionInviteLink, allowing bots to create subscription invite links.
  - Added the method aiogram.methods.edit\_chat\_subscription\_invite\_link.
     EditChatSubscriptionInviteLink, allowing bots to edit the name of subscription invite links.
  - Added the field until\_date to the class aiogram.types.chat\_member\_member.ChatMemberMember for members with an active subscription.
  - Added support for paid reactions and the class aiogram.types.reaction\_type\_paid.
     ReactionTypePaid.

#1560

#### Misc

• Improved performance of StatesGroup #1507

#### 2.6.4 3.11.0 (2024-08-09)

#### **Features**

- Added full support of Bot API 7.8
  - Added the field has\_main\_web\_app to the class aiogram.types.user.User, which is returned in the response to aiogram.methods.get\_me.GetMe.
  - Added the parameter business\_connection\_id to the methods aiogram.methods.
     pin\_chat\_message.PinChatMessage and aiogram.methods.unpin\_chat\_message.
     UnpinChatMessage, allowing bots to manage pinned messages on behalf of a business account.

#1551

# **Bugfixes**

• Fixed URL path in the "Open" button at the "demo/sendMessage" endpoint in the web\_app example. #1546

#### Misc

- Added method aiogram.types.message.Message.as\_reply\_parameters(). Replaced usage of the argument reply\_to\_message\_id with reply\_parameters in all Message reply methods. #1538
- Added aiohttp v3.10 \ support. #1548

# 2.6.5 3.10.0 (2024-07-07)

#### **Features**

- Added full support of Bot API 7.7
  - Added the class aiogram.types.refunded\_payment.RefundedPayment, containing information about a refunded payment.
  - Added the field refunded\_payment to the class aiogram.types.message.Message, describing a service message about a refunded payment.

#1536

# 2.6.6 3.9.0 (2024-07-06)

#### **Features**

- Added ChatMember resolution tool and updated 2.x migration guide. #1525
- Added full support of Bot API 7.6
  - Added the classes aiogram.types.paid\_media.PaidMedia,

aiogram.types.paid\_media\_info.PaidMediaInfo, aiogram.types.paid\_media\_preview. PaidMediaPreview, aiogram.types.paid\_media\_photo.PaidMediaPhoto and aiogram.types.paid\_media\_video.PaidMediaVideo, containing information about paid media.

- Added the method aiogram.methods.send\_paid\_media.SendPaidMedia

and the classes aiogram.types.input\_paid\_media.InputPaidMedia, aiogram.types.input\_paid\_media\_photo.InputPaidMediaPhoto and aiogram.types.input\_paid\_media\_video.InputPaidMediaVideo, to support sending paid media.

- Documented that the methods aiogram.methods.copy\_message.CopyMessage
   and aiogram.methods.copy\_messages.CopyMessages cannot be used to copy paid media.
- Added the field can\_send\_paid\_media to the class aiogram.types.chat\_full\_info.ChatFullInfo.
- Added the field paid\_media to the classes

aiogram.types.message.Message and aiogram.types.external\_reply\_info.

- Added the class

aiogram.types.transaction\_partner\_telegram\_ads.TransactionPartnerTelegramAds, containing information about Telegram Star transactions involving the Telegram Ads Platform.

2.6. Changelog 677

- Added the field invoice\_payload to the class

aiogram.types.transaction\_partner\_user.TransactionPartnerUser, containing the bot-specified invoice payload.

- Changed the default opening mode for Direct Link Mini Apps.
- Added support for launching Web Apps via t.me link in the class

```
aiogram.types.menu_button_web_app.MenuButtonWebApp.
```

- Added the field section\_separator\_color to the class ThemeParams.

#1533

## **Bugfixes**

Fixed event context resolving for the callback query that is coming from the business account #1520

# 2.6.7 3.8.0 (2024-06-19)

#### **Features**

- Added utility to safely deserialize any Telegram object or method to a JSON-compatible object (dict). (>> Read more) #1450
- Added full support of Bot API 7.5
  - Added the classes aiogram.types.star\_transactions.StarTransactions,

```
aiogram.types.star_transaction.StarTransaction, aiogram.types.transaction_partner.TransactionPartner and aiogram.types.revenue_withdrawal_state.RevenueWithdrawalState, containing information about Telegram Star transactions involving the bot.
```

- Added the method aiogram.methods.get\_star\_transactions.GetStarTransactions
  that can be used to get the list of all Telegram Star transactions for the bot.
- Added support for callback buttons in

aiogram.types.inline\_keyboard\_markup.InlineKeyboardMarkup for messages sent on behalf of a business account.

- Added support for callback queries originating from a message sent

on behalf of a business account.

Added the parameter business\_connection\_id to the methods

```
aiogram.methods.edit_message_text.EditMessageText, aiogram.methods.edit_message_media.EditMessageMedia, aiogram.methods.edit_message_caption.

EditMessageCaption, aiogram.methods.edit_message_live_location.

EditMessageLiveLocation, aiogram.methods.stop_message_live_location.

StopMessageLiveLocation and aiogram.methods.edit_message_reply_markup.

EditMessageReplyMarkup, allowing the bot to edit business messages.
```

- Added the parameter business\_connection\_id to the method

aiogram.methods.stop\_poll.StopPoll, allowing the bot to stop polls it sent on behalf of a business account.

#1518

### **Bugfixes**

Increased DNS cache ttl setting to aiohttp session as a workaround for DNS resolution issues in aiohttp. #1500

### **Improved Documentation**

- Fixed MongoStorage section in the documentation by adding extra dependency to ReadTheDocs configuration.
   #1501
- Added information about dependency changes to the 2.x --> 3.x migration guide. #1504

#### Misc

- [Only for contributors] Fail redis and mongo tests if incorrect URI provided + some storages tests refactoring If incorrect URIs provided to "-redis" and/or "-mongo" options tests should fail with errors instead of skipping. Otherwise the next scenario is possible:
  - 1) developer breaks RedisStorage and/or MongoStorage code
  - 2) tests are run with incorrect redis and/or mongo URIsprovided by "-redis" and "-mongo" options (for example, wrong port specified)
  - 3) tests pass because skipping doesn't fail tests run
  - 4) developer or reviewer doesn't notice that redis and/or mongo tests were skipped
  - 5) broken code gets in codebase

Also some refactorings done (related with storages and storages tests). #1510

### 2.6.8 3.7.0 (2024-05-31)

#### **Features**

- Added new storage aiogram.fsm.storage.MongoStorage for Finite State Machine based on Mongo DB (using motor library) #1434
- Added full support of Bot API 7.4 #1498

### **Bugfixes**

• Fixed wrong MarkdownV2 custom emoji parsing in aiogram.utils.text\_decorations #1496

#### **Deprecations and Removals**

• Removed deprecated arguments from Bot class parse\_mode, disable\_web\_page\_preview, protect\_content as previously announced in v3.4.0. #1494

#### **Misc**

• Improved code consistency and readability in code examples by refactoring imports, adjusting the base webhook URL, modifying bot instance initialization to utilize DefaultBotProperties, and updating router message handlers. #1482

## 2.6.9 3.6.0 (2024-05-06)

#### **Features**

• Added full support of Bot API 7.3 #1480

### **Improved Documentation**

• Added telegram objects transformation block in 2.x -> 3.x migration guide #1412

## 2.6.10 3.5.0 (2024-04-23)

#### **Features**

- Added message\_thread\_id parameter to ChatActionSender class methods. #1437
- Added context manager interface to Bot instance, from now you can use:

```
async with Bot(...) as bot:
```

instead of

```
async with Bot(...).context() as bot:
...
```

#1468

### **Bugfixes**

- **WebAppUser Class Fields**: Added missing *is\_premium*, *added\_to\_attachment\_menu*, and *allows\_write\_to\_pm* fields to *WebAppUser* class to align with the Telegram API.
  - WebAppChat Class Implementation: Introduced the WebAppChat class with all its fields (id, type, title, username, and photo\_url) as specified in the Telegram API, which was previously missing from the library.
  - WebAppInitData Class Fields: Included previously omitted fields in the WebAppInitData class: chat, chat\_type, chat\_instance, to match the official documentation for a complete Telegram Web Apps support.

#1424

- Fixed poll answer FSM context by handling voter\_chat for poll\_answer event #1436
- Added missing error handling to \_background\_feed\_update (when in handle\_in\_background=True web-hook mode) #1458

### **Improved Documentation**

Added WebAppChat class to WebApp docs, updated uk\_UA localisation of WebApp docs. #1433

#### Misc

- Added full support of Bot API 7.2 #1444
- Loosened pydantic version upper restriction from <2.7 to <2.8 #1460

### 2.6.11 3.4.1 (2024-02-17)

#### **Bugfixes**

• Fixed JSON serialization of the LinkPreviewOptions class while it is passed as bot-wide default options. #1418

### 2.6.12 3.4.0 (2024-02-16)

#### **Features**

• Reworked bot-wide globals like parse\_mode, disable\_web\_page\_preview, and others to be more flexible.



### Warning

Note that the old way of setting these global bot properties is now deprecated and will be removed in the next major release.

#### #1392

- A new enum KeyboardButtonPollTypeType for KeyboardButtonPollTypeType.type field has bed added. #1398
- Added full support of Bot API 7.1
  - Added support for the administrator rights can\_post\_stories, can\_edit\_stories, can\_delete\_stories in supergroups.
  - Added the class ChatBoostAdded and the field boost\_added to the class Message for service messages about a user boosting a chat.
  - Added the field sender\_boost\_count to the class Message.
  - Added the field reply\_to\_story to the class Message.
  - Added the fields chat and id to the class Story.
  - Added the field unrestrict\_boost\_count to the class Chat.
  - Added the field custom\_emoji\_sticker\_set\_name to the class Chat.

#1417

### **Bugfixes**

- Update KeyboardBuilder utility, fixed type-hints for button method, adjusted limits of the different markup types to real world values. #1399
- Added new reply\_parameters param to message.send\_copy because it hasn't been added there #1403

## **Improved Documentation**

• Add notion "Working with plural forms" in documentation Utils -> Translation #1395

## 2.6.13 3.3.0 (2023-12-31)

#### **Features**

- Added full support of Bot API 7.0
  - Reactions
  - Replies 2.0
  - Link Preview Customization
  - Block Quotation
  - Multiple Message Actions
  - Requests for multiple users
  - Chat Boosts
  - Giveaway
  - Other changes

#1387

## 2.6.14 3.2.0 (2023-11-24)

#### **Features**

- Introduced Scenes feature that helps you to simplify user interactions using Finite State Machine. Read more about Scenes. #1280
- Added the new FSM strategy CHAT\_TOPIC, which sets the state for the entire topic in the chat, also works in private messages and regular groups without topics. #1343

### **Bugfixes**

- Fixed parse\_mode argument in the in Message.send\_copy shortcut. Disable by default. #1332
- Added ability to get handler flags from filters. #1360
- Fixed a situation where a CallbackData could not be parsed without a default value. #1368

#### **Improved Documentation**

- Corrected grammatical errors, improved sentence structures, translation for migration 2.x-3.x #1302
- Minor typo correction, specifically in module naming + some grammar. #1340
- Added *CITATION.cff* file for automatic academic citation generation. Now you can copy citation from the GitHub page and paste it into your paper. #1351
- Minor typo correction in middleware docs. #1353

#### **Misc**

- Fixed ResourceWarning in the tests, reworked RedisEventsIsolation fixture to use Redis connection from RedisStorage #1320
- Updated dependencies, bumped minimum required version:
  - magic-filter fixed .resolve operation
  - pydantic fixed compatibility (broken in 2.4)
  - aiodns added new dependency to the fast extras (pip install aiogram[fast])
  - others...

#1327

- Prevent update handling task pointers from being garbage collected, backport from 2.x #1331
- Updated typing-extensions package version range in dependencies to fix compatibility with FastAPI #1347
- Introduce Python 3.12 support #1354
- Speeded up CallableMixin processing by caching references to nested objects and simplifying kwargs assembly.
   #1357
- Added pydantic v2.5 support. #1361
- Updated thumbnail fields type to InputFile only #1372

### 2.6.15 3.1.1 (2023-09-25)

### **Bugfixes**

• Fixed *pydantic* version <2.4, since 2.4 has breaking changes. #1322

## 2.6.16 3.1.0 (2023-09-22)

#### **Features**

- Added support for custom encoders/decoders for payload (and also for deep-linking). #1262
- Added aiogram.utils.input\_media.MediaGroupBuilder for media group construction. #1293
- Added full support of Bot API 6.9 #1319

### **Bugfixes**

- Added actual param hints for InlineKeyboardBuilder and ReplyKeyboardBuilder. #1303
- Fixed priority of events isolation, now user state will be loaded only after lock is acquired #1317

## 2.6.17 3.0.0 (2023-09-01)

### **Bugfixes**

- Replaced datetime.datetime with DateTime type wrapper across types to make dumped JSONs object more compatible with data that is sent by Telegram. #1277
- Fixed magic .as\_(...) operation for values that can be interpreted as False (e.g. 0). #1281
- Italic markdown from utils now uses correct decorators #1282
- Fixed method Message.send\_copy for stickers. #1284
- Fixed Message.send\_copy method, which was not working properly with stories, so not you can copy stories too (forwards messages). #1286
- Fixed error overlapping when validation error is caused by remove\_unset root validator in base types and methods.
   #1290

### 2.6.18 3.0.0rc2 (2023-08-18)

#### **Bugfixes**

- Fixed missing message content types (ContentType.USER\_SHARED, ContentType.CHAT\_SHARED) #1252
- Fixed nested hashtag, cashtag and email message entities not being parsed correctly when these entities are inside another entity. #1259
- Moved global filters check placement into router to add chance to pass context from global filters into handlers in the same way as it possible in other places #1266

### **Improved Documentation**

- Added error handling example examples/error\_handling.py #1099
- Added a few words about skipping pending updates #1251
- Added a section on Dependency Injection technology #1253
- This update includes the addition of a multi-file bot example to the repository. #1254
- Refactored examples code to use aiogram enumerations and enhanced chat messages with markdown beautification's for a more user-friendly display. #1256
- Supplemented "Finite State Machine" section in Migration FAQ #1264
- Removed extra param in docstring of TelegramEventObserver's filter method and fixed typo in I18n documentation. #1268

#### Misc

- Enhanced the warning message in dispatcher to include a JSON dump of the update when update type is not known. #1269
- Added support for Bot API 6.8 #1275

## 2.6.19 3.0.0rc1 (2023-08-06)

#### **Features**

• Added Currency enum. You can use it like this:

```
from aiogram.enums import Currency
await bot.send_invoice(
    ...,
    currency=Currency.USD,
    ...
)
```

### #1194

- Updated keyboard builders with new methods for integrating buttons and keyboard creation more seamlessly.
   Added functionality to create buttons from existing markup and attach another builder. This improvement aims to make the keyboard building process more user-friendly and flexible. #1236
- Added support for message\_thread\_id in ChatActionSender #1249

## **Bugfixes**

- Fixed polling startup when "bot" key is passed manually into dispatcher workflow data #1242
- Added codegen configuration for lost shortcuts:
  - ShippingQuery.answer
  - PreCheckoutQuery.answer
  - Message.delete\_reply\_markup

#1244

### **Improved Documentation**

• Added documentation for webhook and polling modes. #1241

#### Misc

- Reworked InputFile reading, removed \_\_aiter\_\_ method, added *bot: Bot* argument to the .read(...) method, so, from now URLInputFile can be used without specifying bot instance. #1238
- Code-generated \_\_init\_\_ typehints in types and methods to make IDE happy without additional pydantic plugin #1245

## 2.6.20 3.0.0b9 (2023-07-30)

#### **Features**

- Added new shortcuts for aiogram.types.chat\_member\_updated.ChatMemberUpdated to send message to chat that member joined/left. #1234
- Added new shortcuts for aiogram.types.chat\_join\_request.ChatJoinRequest to make easier access to sending messages to users who wants to join to chat. #1235

#### **Bugfixes**

- Fixed bot assignment in the Message.send\_copy shortcut #1232
- Added model validation to remove UNSET before field validation. This change was necessary to correctly handle
  parse\_mode where 'UNSET' is used as a sentinel value. Without the removal of 'UNSET', it would create issues
  when passed to model initialization from Bot.method\_name. 'UNSET' was also added to typing. #1233
- Updated pydantic to 2.1 with few bugfixes

### **Improved Documentation**

• Improved docs, added basic migration guide (will be expanded later) #1143

### **Deprecations and Removals**

• Removed the use of the context instance (Bot.get\_current) from all placements that were used previously. This is to avoid the use of the context instance in the wrong place. #1230

### 2.6.21 3.0.0b8 (2023-07-17)

#### **Features**

- Added possibility to use custom events in routers (If router does not support custom event it does not break and passes it to included routers). #1147
- Added support for FSM in Forum topics.

The strategy can be changed in dispatcher:

```
from aiogram.fsm.strategy import FSMStrategy
...
dispatcher = Dispatcher(
    fsm_strategy=FSMStrategy.USER_IN_TOPIC,
    storage=..., # Any persistent storage
)
```

#### 1 Note

If you have implemented you own storages you should extend record key generation with new one attribute - thread\_id

### #1161

- Improved CallbackData serialization.
  - Minimized UUID (hex without dashes)
  - Replaced bool values with int (true=1, false=0)

#### #1163

- Added a tool to make text formatting flexible and easy. More details on the corresponding documentation page #1172
- Added X-Telegram-Bot-Api-Secret-Token header check #1173
- Made allowed\_updates list to revolve automatically in start\_polling method if not set explicitly. #1178
- Added possibility to pass custom headers to URLInputFile object #1191

### **Bugfixes**

· Change type of result in InlineQueryResult enum for InlineQueryResultCachedMpeg4Gif and InlineQueryResultMpeg4Gif to more correct according to documentation.

Change regexp for entities parsing to more correct (InlineQueryResultType.yml). #1146

- Fixed signature of startup/shutdown events to include the \*\*dispatcher.workflow\_data as the handler arguments. #1155
- Added missing FORUM\_TOPIC\_EDITED value to content\_type property #1160
- Fixed compatibility with Python 3.8-3.9 (from previous release) #1162
- Fixed the markdown spoiler parser. #1176
- Fixed workflow data propagation #1196
- Fixed the serialization error associated with nested subtypes like InputMedia, ChatMember, etc.

The previously generated code resulted in an invalid schema under pydantic v2, which has stricter type parsing. Hence, subtypes without the specification of all subtype unions were generating an empty object. This has been rectified now. #1213

### **Improved Documentation**

Changed small grammar typos for upload\_file #1133

### **Deprecations and Removals**

• Removed text filter in due to is planned to remove this filter few versions ago.

Use F. text instead #1170

#### Misc

• Added full support of Bot API 6.6



### **Danger**

Note that this issue has breaking changes described in the Bot API changelog, this changes is not breaking in the API but breaking inside aiogram because Beta stage is not finished.

#### #1139

• Added full support of Bot API 6.7



#### Warning

Note that arguments switch\_pm\_parameter and switch\_pm\_text was deprecated and should be changed to button argument as described in API docs.

#1168

• Updated Pydantic to V2



### Warning

Be careful, not all libraries is already updated to using V2

#1202

- · Added global defaults disable\_web\_page\_preview and protect\_content in addition to parse\_mode to the Bot instance, reworked internal request builder mechanism. #1142
- Removed bot parameters from storages #1144
- Replaced ContextVar's with a new feature called Validation Context in Pydantic to improve the clarity, usability, and versatility of handling the Bot instance within method shortcuts.



**Breaking**: The 'bot' argument now is required in *URLInputFile* 

#1210

- Updated magic-filter with new features
  - Added hint for len(F) error
  - Added not in operation

#1221

## 2.6.22 3.0.0b7 (2023-02-18)

### Warning

Note that this version has incompatibility with Python 3.8-3.9 in case when you create an instance of Dispatcher outside of the any coroutine.

Sorry for the inconvenience, it will be fixed in the next version.

This code will not work:

```
dp = Dispatcher()
def main():
    dp.run_polling(...)
main()
```

But if you change it like this it should works as well:

```
router = Router()
async def main():
   dp = Dispatcher()
    dp.include_router(router)
   dp.start_polling(...)
```

#### 2.6. Changelog

689

#### **Features**

Added missing shortcuts, new enums, reworked old stuff

Breaking All previously added enums is re-generated in new place - aiogram.enums instead of aiogram.types

```
Added enums: aiogram.enums.bot_command_scope_type.BotCommandScopeType,
```

```
aiogram.enums.chat_action.ChatAction.
                                                aiogram.enums.chat_member_status.
ChatMemberStatus,
                         aiogram.enums.chat_type.ChatType,
                                                                   aiogram.enums.
content_type.ContentType,
                            aiogram.enums.dice_emoji.DiceEmoji,
                                                                   aiogram.enums.
inline_query_result_type.InlineQueryResultType,
                                                  aiogram.enums.input_media_type.
                 aiogram.enums.mask_position_point.MaskPositionPoint,
InputMediaType,
                                                                         aiogram.
enums.menu_button_type.MenuButtonType,
                                               aiogram.enums.message_entity_type.
MessageEntityType, aiogram.enums.parse_mode.ParseMode, aiogram.enums.poll_type.
PollType, aiogram.enums.sticker_type.StickerType, aiogram.enums.topic_icon_color.
TopicIconColor, aiogram.enums.update_type.UpdateType,
```

#### Added shortcuts:

- Chat aiogram.types.chat.Chat.get\_administrators(),

```
aiogram.types.chat.Chat.delete_message(),
                                                     aiogram.types.chat.Chat.
revoke_invite_link(),
                                  aiogram.types.chat.Chat.edit_invite_link(),
aiogram.types.chat.Chat.create_invite_link(),
                                                     aiogram.types.chat.Chat.
export_invite_link(), aiogram.types.chat.Chat.do(), aiogram.types.chat.Chat.
delete_sticker_set(), aiogram.types.chat.Chat.set_sticker_set(), aiogram.types.
chat.Chat.get_member(), aiogram.types.chat.Chat.get_member_count(), aiogram.
types.chat.Chat.leave(), aiogram.types.chat.Chat.unpin_all_messages(), aiogram.
types.chat.Chat.unpin_message(),
                                       aiogram.types.chat.Chat.pin_message(),
aiogram.types.chat.Set_administrator_custom_title(), aiogram.types.chat.
Chat.set_permissions(), aiogram.types.chat.Chat.promote(), aiogram.types.chat.
Chat.restrict(),
                  aiogram.types.chat.Chat.unban(), aiogram.types.chat.Chat.
        aiogram.types.chat.Chat.set_description(),
                                                     aiogram.tvpes.chat.Chat.
set_title(), aiogram.types.chat.Chat.delete_photo(), aiogram.types.chat.Chat.
set_photo(),
```

- Sticker: aiogram.types.sticker.Sticker.set\_position\_in\_set(), aiogram.types.sticker.Sticker.delete\_from\_set(),
- User: aiogram.types.user.User.get\_profile\_photos()

#952

- Added callback answer feature #1091
- Added a method that allows you to compactly register routers #1117

### **Bugfixes**

- Check status code when downloading file #816
- Fixed ignore\_case parameter in aiogram. filters.command.Command filter #1106

#### **Misc**

- Added integration with new code-generator named Butcher #1069
- Added full support of Bot API 6.4 #1088
- Updated package metadata, moved build internals from Poetry to Hatch, added contributing guides. #1095
- Added full support of Bot API 6.5

## **Danger**

Note that aiogram.types.chat\_permissions.ChatPermissions is updated without backward compatibility, so now this object has no can\_send\_media\_messages attribute

#### #1112

- Replaced error TypeError: TelegramEventObserver.\_\_call\_\_() got an unexpected keyword argument '<name>' with a more understandable one for developers and with a link to the documentation. #1114
- Added possibility to reply into webhook with files #1120
- Reworked graceful shutdown. Added method to stop polling. Now polling started from dispatcher can be stopped by signals gracefully without errors (on Linux and Mac). #1124

## 2.6.23 3.0.0b6 (2022-11-18)

#### **Features**

#1030

• (again) Added possibility to combine filters with an *and/or* operations.

Read more in "Combining filters" documentation section #1018

Added following methods to Message class:

```
- Message.forward(...)
- Message.edit_media(...)
- Message.edit_live_location(...)
- Message.stop_live_location(...)
- Message.pin(...)
- Message.unpin()
```

• Added following methods to User class:

```
- User.mention_markdown(...)
```

- User.mention\_html(...)

#1049

• Added full support of Bot API 6.3 #1057

### **Bugfixes**

- Fixed Message.send\_invoice and Message.reply\_invoice, added missing arguments #1047
- Fixed copy and forward in:

```
- Message.answer(...)
- Message.copy_to(...)
#1064
```

### **Improved Documentation**

- Fixed UA translations in index.po #1017
- Fix typehints for Message, reply\_media\_group and answer\_media\_group methods #1029
- Removed an old now non-working feature #1060

#### **Misc**

- Enabled testing on Python 3.11 #1044
- Added a mandatory dependency certifi in due to in some cases on systems that doesn't have updated cacertificates the requests to Bot API fails with reason [SSL: CERTIFICATE\_VERIFY\_FAILED] certificate verify failed: self signed certificate in certificate chain #1066

## 2.6.24 3.0.0b5 (2022-10-02)

#### **Features**

- Add PyPy support and run tests under PyPy #985
- Added message text to aiogram exceptions representation #988
- Added warning about using magic filter from *magic\_filter* instead of *aiogram*'s ones. Is recommended to use *from aiogram import F* instead of *from magic\_filter import F* #990
- Added more detailed error when server response can't be deserialized. This feature will help to debug unexpected responses from the Server #1014

### **Bugfixes**

- Reworked error event, introduced aiogram.types.error\_event.ErrorEvent object. #898
- Fixed escaping markdown in aiogram.utils.markdown module #903
- Fixed polling crash when Telegram Bot API raises HTTP 429 status-code. #995
- Fixed empty mention in command parsing, now it will be None instead of an empty string #1013

### **Improved Documentation**

• Initialized Docs translation (added Ukrainian language) #925

### **Deprecations and Removals**

• Removed filters factory as described in corresponding issue. #942

#### **Misc**

• Now Router/Dispatcher accepts only keyword arguments. #982

## 2.6.25 3.0.0b4 (2022-08-14)

#### **Features**

- Add class helper ChatAction for constants that Telegram BotAPI uses in sendChatAction request. In my opinion, this will help users and will also improve compatibility with 2.x version where similar class was called "ChatActions". #803
- · Added possibility to combine filters or invert result

Example:

```
Text(text="demo") | Command(commands=["demo"])
MyFilter() & AnotherFilter()
~StateFilter(state='my-state')
```

#894

- Fixed type hints for redis TTL params. #922
- Added *full\_name* shortcut for *Chat* object #929

#### **Bugfixes**

- Fixed false-positive coercing of Union types in API methods #901
- Added 3 missing content types:
  - proximity\_alert\_triggered
  - supergroup\_chat\_created
  - channel\_chat\_created

#### #906

- Fixed the ability to compare the state, now comparison to copy of the state will return True. #927
- Fixed default lock kwargs in RedisEventIsolation. #972

#### Misc

- Restrict including routers with strings #896
- Changed CommandPatterType to CommandPatternType in aiogram/dispatcher/filters/command.py #907
- Added full support of Bot API 6.1 #936
- Breaking! More flat project structure

These packages was moved, imports in your code should be fixed:

- aiogram.dispatcher.filters -> aiogram.filters
- aiogram.dispatcher.fsm -> aiogram.fsm
- aiogram.dispatcher.handler -> aiogram.handler
- aiogram.dispatcher.webhook -> aiogram.webhook
- aiogram.dispatcher.flags/\* -> aiogram.dispatcher.flags (single module instead of package)

#### #938

- Removed deprecated router.<event>\_handler and router.register\_<event>\_handler methods. #941
- Deprecated filters factory. It will be removed in next Beta (3.0b5) #942
- MessageEntity method get\_text was removed and extract was renamed to extract\_from #944
- Added full support of Bot API 6.2 #975

### 2.6.26 3.0.0b3 (2022-04-19)

#### **Features**

- Added possibility to get command magic result as handler argument #889
- Added full support of Telegram Bot API 6.0 #890

### **Bugfixes**

- Fixed I18n lazy-proxy. Disabled caching. #839
- Added parsing of spoiler message entity #865
- Fixed default parse\_mode for Message.copy\_to() method. #876
- Fixed CallbackData factory parsing IntEnum's #885

#### **Misc**

- Added automated check that pull-request adds a changes description to CHANGES directory #873
- Changed Message.html\_text and Message.md\_text attributes behaviour when message has no text. The
  empty string will be used instead of raising error. #874
- Used redis-py instead of aioredis package in due to this packages was merged into single one #882
- Solved common naming problem with middlewares that confusing too much developers now you can't see
  the middleware and middlewares attributes at the same point because this functionality encapsulated to special
  interface. #883

## 2.6.27 3.0.0b2 (2022-02-19)

### **Features**

- Added possibility to pass additional arguments into the aiohttp webhook handler to use this arguments inside handlers as the same as it possible in polling mode. #785
- Added possibility to add handler flags via decorator (like pytest.mark decorator but aiogram.flags) #836
- Added ChatActionSender utility to automatically sends chat action while long process is running.
   It also can be used as message middleware and can be customized via chat\_action flag. #837

### **Bugfixes**

- Fixed unexpected behavior of sequences in the StateFilter. #791
- Fixed exceptions filters #827

#### **Misc**

- Logger name for processing events is changed to aiogram.events. #830
- Added full support of Telegram Bot API 5.6 and 5.7 #835
- BREAKING Events isolation mechanism is moved from FSM storages to standalone managers #838

### 2.6.28 3.0.0b1 (2021-12-12)

#### **Features**

• Added new custom operation for MagicFilter named as\_

Now you can use it to get magic filter result as handler argument

(continues on next page)

(continued from previous page)

```
@router.message(F.photo[-1].as_("photo"))
async def download_photos_handler(message: Message, photo: PhotoSize, bot: Bot):
    content = await bot.download(photo)
```

#759

### **Bugfixes**

Fixed: Missing ChatMemberHandler import in aiogram/dispatcher/handler #751

### **Misc**

- Check destiny in case of no with\_destiny enabled in RedisStorage key builder #776
- Added full support of Bot API 5.5 #777
- Stop using feature from #336. From now settings of client-session should be placed as initializer arguments instead of changing instance attributes. #778
- Make TelegramAPIServer files wrapper in local mode bi-directional (server-client, client-server) Now you can convert local path to server path and server path to local path. #779

### 2.6.29 3.0.0a18 (2021-11-10)

#### **Features**

- Breaking: Changed the signature of the session middlewares Breaking: Renamed AiohttpSession.make\_request
  method parameter from call to method to match the naming in the base class Added middleware for logging
  outgoing requests #716
- Improved description of filters resolving error. For example when you try to pass wrong type of argument to the filter but don't know why filter is not resolved now you can get error like this:

```
aiogram.exceptions.FiltersResolveError: Unknown keyword filters: {'content_types'}
Possible cases:
- 1 validation error for ContentTypesFilter
    content_types
        Invalid content types {'42'} is not allowed here (type=value_error)
```

#717

- Breaking internal API change Reworked FSM Storage record keys propagation #723
- Implemented new filter named MagicData(magic\_data) that helps to filter event by data from middlewares or other filters

For example your bot is running with argument named config that contains the application config then you can filter event by value from this config:

```
@router.message(magic_data=F.event.from_user.id == F.config.admin_id)
...
```

#724

## **Bugfixes**

- Fixed I18n context inside error handlers #726
- Fixed bot session closing before emit shutdown #734
- Fixed: bound filter resolving does not require children routers #736

#### **Misc**

- Enabled testing on Python 3.10 Removed *async\_lru* dependency (is incompatible with Python 3.10) and replaced usage with protected property #719
- Converted README.md to README.rst and use it as base file for docs #725
- Rework filters resolving:
  - Automatically apply Bound Filters with default values to handlers
  - Fix data transfer from parent to included routers filters

#727

Added full support of Bot API 5.4 https://core.telegram.org/bots/api-changelog#november-5-2021 #744

### 2.6.30 3.0.0a17 (2021-09-24)

#### **Misc**

- Added html\_text and md\_text to Message object #708
- Refactored I18n, added context managers for I18n engine and current locale #709

### 2.6.31 3.0.0a16 (2021-09-22)

#### **Features**

- Added support of local Bot API server files downloading
   When Local API is enabled files can be downloaded via bot.download/bot.download file methods. #698
- Implemented I18n & L10n support #701

#### **Misc**

- Covered by tests and docs KeyboardBuilder util #699
- Breaking!!!. Refactored and renamed exceptions.
  - Exceptions module was moved from aiogram.utils.exceptions to aiogram.exceptions
  - Added prefix *Telegram* for all error classes

#700

• Replaced all pragma: no cover marks via global .coveragerc config #702

• Updated dependencies.

**Breaking for framework developers** Now all optional dependencies should be installed as extra: *poetry install -E fast -E redis -E proxy -E i18n -E docs* #703

### 2.6.32 3.0.0a15 (2021-09-10)

### **Features**

 Ability to iterate over all states in StatesGroup. Aiogram already had in check for states group so this is relative feature. #666

## **Bugfixes**

• Fixed incorrect type checking in the aiogram.utils.keyboard.KeyboardBuilder #674

#### **Misc**

- Disable ContentType filter by default #668
- Moved update type detection from Dispatcher to Update object #669
- Updated **pre-commit** config #681
- Reworked handlers\_in\_use util. Function moved to Router as method .resolve\_used\_update\_types() #682

## 2.6.33 3.0.0a14 (2021-08-17)

#### **Features**

- add aliases for edit/delete reply markup to Message #662
- Reworked outer middleware chain. Prevent to call many times the outer middleware for each nested router #664

### **Bugfixes**

• Prepare parse mode for InputMessageContent in AnswerInlineQuery method #660

### **Improved Documentation**

• Added integration with towncrier #602

#### **Misc**

- Added .editorconfig #650
- Redis storage speedup globals #651
- add allow\_sending\_without\_reply param to Message reply aliases #663

## 2.6.34 2.14.3 (2021-07-21)

• Fixed ChatMember type detection via adding customizable object serialization mechanism (#624, #623)

## 2.6.35 2.14.2 (2021-07-26)

- Fixed MemoryStorage cleaner (#619)
- Fixed unused default locale in I18nMiddleware (#562, #563)

### 2.6.36 2.14 (2021-07-27)

- Full support of Bot API 5.3 (#610, #614)
- Fixed Message.send\_copy method for polls (#603)
- Updated pattern for GroupDeactivated exception (#549
- Added caption\_entities field in InputMedia base class (#583)
- Fixed HTML text decorations for tag pre (#597 fixes issues #596 and #481)
- Fixed Message.get\_full\_command method for messages with caption (#576)
- Improved MongoStorage: remove documents with empty data from aiogram\_data collection to save memory. (#609)

## 2.6.37 2.13 (2021-04-28)

- Added full support of Bot API 5.2 (#572)
- Fixed usage of provider\_data argument in sendInvoice method call
- Fixed builtin command filter args (#556) (#558)
- Allowed to use State instances FSM storage directly (#542)
- Added possibility to get i18n locale without User instance (#546)
- Fixed returning type of Bot.\*\_chat\_invite\_link() methods #548 (#549)
- Fixed deep-linking util (#569)
- Small changes in documentation describe limits in docstrings corresponding to the current limit. (#565)
- Fixed internal call to deprecated 'is\_private' method (#553)
- Added possibility to use allowed\_updates argument in Polling mode (#564)

## 2.6.38 2.12.1 (2021-03-22)

- Fixed TypeError: Value should be instance of 'User' not 'NoneType' (#527)
- Added missing Chat.message\_auto\_delete\_time field (#535)
- Added MediaGroup filter (#528)
- Added Chat.delete\_message shortcut (#526)
- Added mime types parsing for aiogram.types.Document (#431)
- Added warning in TelegramObject.\_\_setitem\_\_ when Telegram adds a new field (#532)
- Fixed examples/chat\_type\_filter.py (#533)
- Removed redundant definitions in framework code (#531)

### 2.6.39 2.12 (2021-03-14)

- Full support for Telegram Bot API 5.1 (#519)
- Fixed sending playlist of audio files and documents (#465, #468)
- Fixed FSMContextProxy.setdefault method (#491)
- Fixed Message.answer\_location and Message.reply\_location unable to send live location (#497)
- Fixed user\_id and chat\_id getters from the context at Dispatcher check\_key, release\_key and throttle methods (#520)
- Fixed Chat.update\_chat method and all similar situations (#516)
- Fixed MediaGroup attach methods (#514)
- Fixed state filter for inline keyboard query callback in groups (#508, #510)
- Added missing ContentTypes.DICE (#466)
- Added missing yeard argument to InputContactMessageContent constructor (#473)
- Add missing exceptions: MessageIdInvalid, CantRestrictChatOwner and UserIsAnAdministratorOfTheChat (#474, #512)
- Added answer\_chat\_action to the Message object (#501)
- Added dice to message.send\_copy method (#511)
- Removed deprecation warning from Message.send\_copy
- Added an example of integration between externally created aiohttp Application and aiogram (#433)
- Added split\_separator argument to safe\_split\_text (#515)
- Fixed some typos in docs and examples (#489, #490, #498, #504, #514)

## 2.6.40 2.11.2 (2021-11-10)

- · Fixed default parse mode
- Added missing "supports\_streaming" argument to answer\_video method #462

## 2.6.41 2.11.1 (2021-11-10)

- Fixed files URL template
- Fix MessageEntity serialization for API calls #457
- When entities are set, default parse\_mode become disabled (#461)
- Added parameter supports\_streaming to reply\_video, remove redundant docstrings (#459)
- Added missing parameter to promoteChatMember alias (#458)

## 2.6.42 2.11 (2021-11-08)

- Added full support of Telegram Bot API 5.0 (#454)
- Added possibility to more easy specify custom API Server (example)
  - WARNING: API method close was named in Bot class as close\_bot in due to Bot instance already
    has method with the same name. It will be changed in aiogram 3.0
- Added alias to Message object Message.copy\_to with deprecation of Message.send\_copy
- ChatType.SUPER\_GROUP renamed to ChatType.SUPERGROUP (#438)

### 2.6.43 2.10.1 (2021-09-14)

• Fixed critical bug with getting asyncio event loop in executor. (#424) AttributeError: 'NoneType' object has no attribute 'run\_until\_complete'

### 2.6.44 2.10 (2021-09-13)

- Breaking change: Stop using \_MainThread event loop in bot/dispatcher instances (#397)
- Breaking change: Replaced aiomongo with motor (#368, #380)
- Fixed: TelegramObject's aren't destroyed after update handling #307 (#371)
- Add setting current context of Telegram types (#369)
- Fixed markdown escaping issues (#363)
- Fixed HTML characters escaping (#409)
- Fixed italic and underline decorations when parse entities to Markdown
- Fixed #413: parse entities positioning (#414)
- Added missing thumb parameter (#362)
- Added public methods to register filters and middlewares (#370)
- Added ChatType builtin filter (#356)

- Fixed IDFilter checking message from channel (#376)
- Added missed answer\_poll and reply\_poll (#384)
- Added possibility to ignore message caption in commands filter (#383)
- · Fixed addStickerToSet method
- Added preparing thumb in send document method (#391)
- Added exception MessageToPinNotFound (#404)
- Fixed handlers parameter-spec solving (#408)
- Fixed CallbackQuery.answer() returns nothing (#420)
- CHOSEN\_INLINE\_RESULT is a correct API-term (#415)
- Fixed missing attributes for Animation class (#422)
- Added missed emoji argument to reply\_dice (#395)
- Added is\_chat\_creator method to ChatMemberStatus (#394)
- Added missed ChatPermissions to \_\_all\_\_ (#393)
- Added is\_forward method to Message (#390)
- Fixed usage of deprecated is\_private function (#421)

and many others documentation and examples changes:

- Updated docstring of RedisStorage2 (#423)
- Updated I18n example (added docs and fixed typos) (#419)
- A little documentation revision (#381)
- Added comments about correct errors\_handlers usage (#398)
- Fixed typo rexex -> regex (#386)
- Fixed docs Quick start page code blocks (#417)
- fixed type hints of callback\_data (#400)
- Prettify readme, update downloads stats badge (#406)

### 2.6.45 2.9.2 (2021-06-13)

- Fixed Message.get\_full\_command() #352
- Fixed markdown util #353

### 2.6.46 2.9 (2021-06-08)

- Added full support of Telegram Bot API 4.9
- Fixed user context at poll\_answer update (#322)
- Fix Chat.set\_description (#325)
- Add lazy session generator (#326)
- Fix text decorations (#315, #316, #328)
- Fix missing InlineQueryResultPhoto parse\_mode field (#331)

- Fix fields from parent object in KeyboardButton (#344 fixes #343)
- Add possibility to get bot id without calling get\_me (#296)

## 2.6.47 2.8 (2021-04-26)

- Added full support of Bot API 4.8
- Added Message.answer\_dice and Message.reply\_dice methods (#306)

## 2.6.48 2.7 (2021-04-07)

- Added full support of Bot API 4.7 (#294 #289)
- Added default parse mode for send\_animation method (#293 #292)
- Added new API exception when poll requested in public chats (#270)
- Make correct User and Chat get\_mention methods (#277)
- · Small changes and other minor improvements

## 2.6.49 2.6.1 (2021-01-25)

- Fixed reply KeyboardButton initializer with request\_poll argument (#266)
- Added helper for poll types (aiogram.types.PollType)
- Changed behavior of Telegram\_object .as\_\* and .to\_\* methods. It will no more mutate the object. (#247)

## 2.6.50 2.6 (2021-01-23)

- Full support of Telegram Bot API v4.6 (Polls 2.0) #265
- Aded new filter IsContactSender (commit)
- Fixed proxy extra dependencies version #262

### 2.6.51 2.5.3 (2021-01-05)

- #255 Updated CallbackData factory validity check. More correct for non-latin symbols
- #256 Fixed renamed\_argument decorator error
- #257 One more fix of CommandStart filter

## 2.6.52 2.5.2 (2021-01-01)

• Get back quote\_html and escape\_md functions

### 2.6.53 2.5.1 (2021-01-01)

• Hot-fix of CommandStart filter

## 2.6.54 2.5 (2021-01-01)

- Added full support of Telegram Bot API 4.5 (#250, #251)
- #239 Fixed check\_token method
- #238, #241: Added deep-linking utils
- #248 Fixed support of aiohttp-socks
- Updated setup.py. No more use of internal pip API
- Updated links to documentations (https://docs.aiogram.dev)
- Other small changes and minor improvements (#223 and others...)

## 2.6.55 2.4 (2021-10-29)

- Added Message.send\_copy method (forward message without forwarding)
- Safe close of aiohttp client session (no more exception when application is shutdown)
- No more "adWanced" words in project #209
- Arguments user and chat is renamed to user\_id and chat\_id in Dispatcher.throttle method #196
- Fixed set\_chat\_permissions #198
- Fixed Dispatcher polling task does not process cancellation #199, #201
- Fixed compatibility with latest asyncio version #200
- Disabled caching by default for lazy\_gettext method of I18nMiddleware #203
- Fixed HTML user mention parser #205
- Added IsReplyFilter #210
- Fixed send\_poll method arguments #211
- Added OrderedHelper #215
- Fix incorrect completion order. #217

## 2.6.56 2.3 (2021-08-16)

- Full support of Telegram Bot API 4.4
- Fixed #143
- Added new filters from issue #151: #172, #176, #182
- Added expire argument to RedisStorage2 and other storage fixes #145
- Fixed JSON and Pickle storages #138
- Implemented MongoStorage #153 based on aiomongo (soon motor will be also added)
- Improved tests
- Updated examples
- Warning: Updated auth widget util. #190
- Implemented throttle decorator #181

### 2.6.57 2.2 (2021-06-09)

- Provides latest Telegram Bot API (4.3)
- Updated docs for filters
- Added opportunity to use different bot tokens from single bot instance (via context manager, #100)
- IMPORTANT: Fixed Typo: data -> bucket in update\_bucket for RedisStorage2 (#132)

### 2.6.58 2.1 (2021-04-18)

- Implemented all new features from Telegram Bot API 4.2
- is\_member and is\_admin methods of ChatMember and ChatMemberStatus was renamed to is chat member and is chat admin
- · Remover func filter
- Added some useful Message edit functions (Message.edit\_caption, Message.edit\_media, Message.edit\_reply\_markup) (#121, #103, #104, #112)
- Added requests timeout for all methods (#110)
- Added answer\* methods to Message object (#112)
- Maked some improvements of CallbackData factory
- Added deep-linking parameter filter to CommandStart filter
- Implemented opportunity to use DNS over socks (#97 -> #98)
- Implemented logging filter for extending LogRecord attributes (Will be usefull with external logs collector utils like GrayLog, Kibana and etc.)
- Updated requirements.txt and dev\_requirements.txt files
- · Other small changes and minor improvements

### 2.6.59 2.0.1 (2021-12-31)

- Implemented CallbackData factory (example)
- Implemented methods for answering to inline query from context and reply with animation to the messages. #85
- Fixed installation from tar.gz #84
- More exceptions (ChatIdIsEmpty and NotEnoughRightsToRestrict)

### 2.6.60 2.0 (2021-10-28)

This update will break backward compability with Python 3.6 and works only with Python 3.7+: - contextvars (PEP-567); - New syntax for annotations (PEP-563).

Changes: - 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();

Full changelog - You can read more details about this release in migration FAQ: https://aiogram.readthedocs.io/en/latest/migration\_1\_to\_2.html

## 2.6.61 1.4 (2021-08-03)

• Bot API 4.0 (#57)

### 2.6.62 1.3.3 (2021-07-16)

- Fixed markup-entities parsing;
- Added more API exceptions;
- Now InlineQueryResultLocation has live\_period;
- Added more message content types;
- · Other small changes and minor improvements.

## 2.6.63 1.3.2 (2021-05-27)

- Fixed crashing of polling process. (i think)
- Added parse mode field into input query results according to Bot API Docs.
- Added new methods for Chat object. (#42, #43)
- Warning: disabled connections limit for bot aiohttp session.
- Warning: Destroyed "temp sessions" mechanism.
- Added new error types.
- Refactored detection of error type.

- Small fixes of executor util.
- · Fixed RethinkDBStorage

### 2.6.64 1.3.1 (2018-05-27)

## 2.6.65 1.3 (2021-04-22)

- Allow to use Socks5 proxy (need manually install aiosocksy).
- Refactored aiogram.utils.executor module.
- [Warning] Updated requirements list.

## 2.6.66 1.2.3 (2018-04-14)

- Fixed API errors detection
- Fixed compability of setup.py with pip 10.0.0

## 2.6.67 1.2.2 (2018-04-08)

- Added more error types.
- Implemented method InputFile.from\_url(url: str) for downloading files.
- Implemented big part of API method tests.
- · Other small changes and mminor improvements.

### 2.6.68 1.2.1 (2018-03-25)

- Fixed handling Venue's [#27, #26]
- Added parse\_mode to all medias (Bot API 3.6 support) [#23]
- Now regexp filter can be used with callback query data [#19]
- Improvements in InlineKeyboardMarkup & ReplyKeyboardMarkup objects [#21]
- Other bug & typo fixes and minor improvements.

### 2.6.69 1.2 (2018-02-23)

- Full provide Telegram Bot API 3.6
- Fixed critical error: Fatal Python error: PyImport\_GetModuleDict: no module dictionary!
- Implemented connection pool in RethinkDB driver
- Typo fixes of documentstion
- · Other bug fixes and minor improvements.

## 2.6.70 1.1 (2018-01-27)

- Added more methods for data types (like message.reply\_sticker(...) or file.download(...)
- Typo fixes of documentation
- Allow to set default parse mode for messages (Bot( ... , parse\_mode='HTML'))
- Allowed to cancel event from the Middleware.on\_pre\_process\_<event type>
- Fixed sending files with correct names.
- · Fixed MediaGroup
- Added RethinkDB storage for FSM (aiogram.contrib.fsm\_storage.rethinkdb)

## 2.6.71 1.0.4 (2018-01-10)

## 2.6.72 1.0.3 (2018-01-07)

- · Added middlewares mechanism.
- · Added example for middlewares and throttling manager.
- Added logging middleware (aiogram.contrib.middlewares.logging.LoggingMiddleware)
- Fixed handling errors in async tasks (marked as 'async\_task')
- Small fixes and other minor improvements.

## 2.6.73 1.0.2 (2017-11-29)

### 2.6.74 1.0.1 (2017-11-21)

- Implemented types.InputFile for more easy sending local files
- Danger! Fixed typo in word pooling. Now whatever all methods with that word marked as deprecated and original methods is renamed to polling. Check it in you'r code before updating!
- Fixed helper for chat actions (types.ChatActions)
- Added example for media group.

### 2.6.75 1.0 (2017-11-19)

- Remaked data types serialozation/deserialization mechanism (Speed up).
- Fully rewrited all Telegram data types.
- Bot object was fully rewritted (regenerated).
- Full provide Telegram Bot API 3.4+ (with sendMediaGroup)
- Warning: Now BaseStorage.close() is awaitable! (FSM)
- Fixed compability with uvloop.
- More employments for aiogram.utils.context.
- Allowed to disable ujson.

- Other bug fixes and minor improvements.
- Migrated from Bitbucket to Github.

```
2.6.76 0.4.1 (2017-08-03)
```

2.6.77 0.4 (2017-08-05)

2.6.78 0.3.4 (2017-08-04)

2.6.79 0.3.3 (2017-07-05)

2.6.80 0.3.2 (2017-07-04)

2.6.81 0.3.1 (2017-07-04)

2.6.82 0.2b1 (2017-06-00)

2.6.83 0.1 (2017-06-03)

## 2.7 Contributing

You're welcome to contribute to aiogram!

aiogram is an open-source project, and anyone can contribute to it in any possible way

## 2.7.1 Developing

Before making any changes in the framework code, it is necessary to fork the project and clone the project to your PC and know how to do a pull-request.

How to work with pull-request you can read in the GitHub docs

Also in due to this project is written in Python, you will need Python to be installed (is recommended to use latest Python versions, but any version starting from 3.8 can be used)

#### Use virtualenv

You can create a virtual environment in a directory using venv module (it should be pre-installed by default):

This action will create a .venv directory with the Python binaries and then you will be able to install packages into that isolated environment.

2.7. Contributing 709

### **Activate the environment**

Linux / macOS:

```
source .venv/bin/activate
```

Windows cmd

```
.\.venv\Scripts\activate
```

Windows PowerShell

```
.\.venv\Scripts\activate.ps1
```

To check it worked, use described command, it should show the pip version and location inside the isolated environment

```
pip -V
```

Also make sure you have the latest pip version in your virtual environment to avoid errors on next steps:

```
python -m pip install --upgrade pip
```

### Setup project

After activating the environment install *aiogram* from sources and their dependencies.

Linux / macOS:

```
pip install -e ."[dev,test,docs,fast,redis,mongo,proxy,i18n]"
```

Windows:

```
pip install -e .[dev,test,docs,fast,redis,mongo,proxy,i18n]
```

It will install aiogram in editable mode into your virtual environment and all dependencies.

### Making changes in code

At this point you can make any changes in the code that you want, it can be any fixes, implementing new features or experimenting.

#### Format the code (code-style)

Note that this project is Black-formatted, so you should follow that code-style, too be sure You're correctly doing this let's reformat the code automatically:

```
black aiogram tests examples isort aiogram tests examples
```

#### **Run tests**

All changes should be tested:

```
pytest tests
```

Also if you are doing something with Redis-storage or/and MongoDB-storage, you will need to test everything works with Redis or/and MongoDB:

#### **Docs**

We are using *Sphinx* to render docs in different languages, all sources located in *docs* directory, you can change the sources and to test it you can start live-preview server and look what you are doing:

```
sphinx-autobuild --watch aiogram/ docs/ docs/_build/
```

#### **Docs translations**

Translation of the documentation is very necessary and cannot be done without the help of the community from all over the world, so you are welcome to translate the documentation into different languages.

Before start, let's up to date all texts:

```
cd docs
make gettext
sphinx-intl update -p _build/gettext -l <language_code>
```

Change the <language\_code> in example below to the target language code, after that you can modify texts inside docs/locale/<language\_code>/LC\_MESSAGES as \*.po files by using any text-editor or specialized utilities for GNU Gettext, for example via poedit.

To view results:

```
sphinx-autobuild --watch aiogram/ docs/ docs/_build/ -D language=<language_code>
```

#### **Describe changes**

Describe your changes in one or more sentences so that bot developers know what's changed in their favorite framework - create <*code*>.<*category*>.*rst* file and write the description.

<code> is Issue or Pull-request number, after release link to this issue will be published to the *Changelog* page.

<category> is a changes category marker, it can be one of:

- feature when you are implementing new feature
- bugfix when you fix a bug
- doc when you improve the docs
- removal when you remove something from the framework
- misc when changed something inside the Core or project configuration

2.7. Contributing 711

If you have troubles with changing category feel free to ask Core-contributors to help with choosing it.

## Complete

After you have made all your changes, publish them to the repository and create a pull request as mentioned at the beginning of the article and wait for a review of these changes.

### 2.7.2 Star on GitHub

You can "star" repository on GitHub - https://github.com/aiogram/aiogram (click the star button at the top right)

Adding stars makes it easier for other people to find this project and understand how useful it is.

### **2.7.3 Guides**

You can write guides how to develop Bots on top of aiogram and publish it into YouTube, Medium, GitHub Books, any Courses platform or any other platform that you know.

This will help more people learn about the framework and learn how to use it

### 2.7.4 Take answers

The developers is always asks for any question in our chats or any other platforms like GitHub Discussions, StackOver-flow and others, feel free to answer to this questions.

## 2.7.5 Funding

The development of the project is free and not financed by commercial organizations, it is my personal initiative (@JRootJunior) and I am engaged in the development of the project in my free time.

So, if you want to financially support the project, or, for example, give me a pizza or a beer, you can do it on OpenCollective.

# **PYTHON MODULE INDEX**

а	<pre>aiogram.methods.approve_chat_join_request,</pre>
aiogram.dispatcher.flags,628	385
aiogram.enums.bot_command_scope_type, 547	aiogram.methods.ban_chat_member,386
aiogram.enums.chat_action, 547	aiogram.methods.ban_chat_sender_chat, 388
aiogram.enums.chat_boost_source_type,548	aiogram.methods.close, 389
aiogram.enums.chat_member_status,548	aiogram.methods.close_forum_topic,390
aiogram.enums.chat_type, 549	<pre>aiogram.methods.close_general_forum_topic,</pre>
aiogram.enums.content_type, 549	391
aiogram.enums.currency, 551	aiogram.methods.copy_message,392
aiogram.enums.dice_emoji,554	aiogram.methods.copy_messages,394
aiogram.enums.encrypted_passport_element,554	<pre>aiogram.methods.create_chat_invite_link, 396</pre>
aiogram.enums.inline_query_result_type, 555	$\verb"aiogram.methods.create_chat_subscription_invite\_link",$
aiogram.enums.input_media_type,556	398
aiogram.enums.input_paid_media_type, 556	aiogram.methods.create_forum_topic,399
aiogram.enums.keyboard_button_poll_type_type,	<pre>aiogram.methods.create_invoice_link,533</pre>
556	<pre>aiogram.methods.create_new_sticker_set,366</pre>
<pre>aiogram.enums.mask_position_point,557</pre>	<pre>aiogram.methods.decline_chat_join_request,</pre>
<pre>aiogram.enums.menu_button_type, 557</pre>	400
<pre>aiogram.enums.message_entity_type,557</pre>	aiogram.methods.delete_chat_photo, 402
aiogram.enums.message_origin_type,558	aiogram.methods.delete_chat_sticker_set, 403
aiogram.enums.paid_media_type,558	aiogram.methods.delete_forum_topic, 404
aiogram.enums.parse_mode,559	aiogram.methods.delete_message,507
<pre>aiogram.enums.passport_element_error_type,</pre>	aiogram.methods.delete_messages, 508
559	aiogram.methods.delete_my_commands, 405
aiogram.enums.poll_type,559	aiogram.methods.delete_sticker_from_set, 368
aiogram.enums.reaction_type_type,560	aiogram.methods.delete_sticker_set, 369
<pre>aiogram.enums.revenue_withdrawal_state_type,</pre>	aiogram.methods.delete_webhook, 540
560	aiogram.methods.edit_chat_invite_link, 406
aiogram.enums.sticker_format,560	aiogram.methods.edit_chat_subscription_invite_link, 408
aiogram.enums.sticker_type, 561	aiogram.methods.edit_forum_topic, 409
aiogram.enums.topic_icon_color,561	aiogram.methods.edit_forum_topic,410
aiogram.enums.transaction_partner_type,561	aiogram.methods.edit_general_forum_topic,410 aiogram.methods.edit_message_caption,509
aiogram.enums.update_type,562	aiogram.methods.edit_message_live_location,
aiogram.exceptions, 626	511
aiogram.handlers.callback_query, 639	aiogram.methods.edit_message_media,513
aiogram.methods.add_sticker_to_set,365	aiogram.methods.edit_message_reply_markup,
aiogram.methods.answer_callback_query, 384	515
aiogram.methods.answer_inline_query, 521	aiogram.methods.edit_message_text,516
aiogram.methods.answer_pre_checkout_query,	aiogram.methods.export_chat_invite_link, 411
530	aiogram.methods.forward_message, 412
<pre>aiogram.methods.answer_shipping_query, 531 aiogram.methods.answer_web_app_query, 524</pre>	aiogram.methods.forward_messages, 414
arogram.methous.answer_web_app_query, 524	alogiam. me chodo i roz nara_meddaged, i i i

```
aiogram.methods.send_video_note, 479
aiogram.methods.get_business_connection, 415
aiogram.methods.get_chat, 416
                                               aiogram.methods.send_voice, 481
aiogram.methods.get_chat_administrators, 417
                                               aiogram.methods.set_chat_administrator_custom_title,
aiogram.methods.get_chat_member,418
aiogram.methods.get_chat_member_count, 419
                                               aiogram.methods.set_chat_description, 485
aiogram.methods.get_chat_menu_button, 420
                                               aiogram.methods.set_chat_menu_button, 486
aiogram.methods.get_custom_emoji_stickers,
                                               aiogram.methods.set_chat_permissions, 487
                                               aiogram.methods.set_chat_photo, 489
aiogram.methods.get_file, 421
                                               aiogram.methods.set_chat_sticker_set, 490
aiogram.methods.get_forum_topic_icon_stickers,aiogram.methods.set_chat_title,491
                                               aiogram.methods.set_custom_emoji_sticker_set_thumbnail,
aiogram.methods.get_game_high_scores, 525
                                                       375
aiogram.methods.get_me, 423
                                               aiogram.methods.set_game_score, 529
aiogram.methods.get_my_commands, 424
                                               aiogram.methods.set_message_reaction, 492
aiogram.methods.get_my_default_administrator_raiphgram.methods.set_my_commands, 493
        425
                                               aiogram.methods.set_my_default_administrator_rights,
aiogram.methods.get_my_description, 426
aiogram.methods.get_my_name, 427
                                               aiogram.methods.set_my_description, 496
aiogram.methods.get_my_short_description, 428
                                               aiogram.methods.set_my_name, 497
aiogram.methods.get_star_transactions, 535
                                               aiogram.methods.set_my_short_description, 498
aiogram.methods.get_sticker_set, 371
                                               aiogram.methods.set_passport_data_errors, 545
aiogram.methods.get_updates, 541
                                               aiogram.methods.set_sticker_emoji_list, 376
aiogram.methods.get_user_chat_boosts, 428
                                               aiogram.methods.set_sticker_keywords, 377
aiogram.methods.get_user_profile_photos, 429
                                               aiogram.methods.set_sticker_mask_position,
aiogram.methods.get_webhook_info,543
aiogram.methods.hide_general_forum_topic, 430
                                               aiogram.methods.set_sticker_position_in_set,
aiogram.methods.leave_chat, 431
aiogram.methods.log_out, 432
                                               aiogram.methods.set_sticker_set_thumbnail,
aiogram.methods.pin_chat_message, 433
                                               aiogram.methods.set_sticker_set_title, 382
aiogram.methods.promote_chat_member, 434
aiogram.methods.refund_star_payment, 536
                                               aiogram.methods.set_webhook, 543
aiogram.methods.reopen_forum_topic, 437
                                               aiogram.methods.stop_message_live_location,
aiogram.methods.reopen_general_forum_topic,
                                               aiogram.methods.stop_poll,520
                                               aiogram.methods.unban_chat_member,499
aiogram.methods.replace_sticker_in_set, 372
aiogram.methods.restrict_chat_member, 439
                                               aiogram.methods.unban_chat_sender_chat, 500
aiogram.methods.revoke_chat_invite_link, 440
                                               aiogram.methods.unhide_general_forum_topic,
aiogram.methods.send_animation, 442
aiogram.methods.send_audio,445
                                               aiogram.methods.unpin_all_chat_messages, 502
aiogram.methods.send_chat_action, 447
                                               aiogram.methods.unpin_all_forum_topic_messages,
aiogram.methods.send_contact, 449
aiogram.methods.send_dice, 451
                                               aiogram.methods.unpin_all_general_forum_topic_messages,
aiogram.methods.send_document, 453
aiogram.methods.send_game, 526
                                               \verb"aiogram.methods.unpin_chat_message", 505
aiogram.methods.send_invoice, 537
                                               aiogram.methods.upload_sticker_file, 383
aiogram.methods.send_location, 456
                                               aiogram.types.animation, 19
                                               aiogram.types.audio, 20
aiogram.methods.send_media_group, 459
aiogram.methods.send_message, 461
                                               aiogram.types.background_fill, 21
                                               aiogram.types.background_fill_freeform_gradient,
aiogram.methods.send_paid_media, 464
aiogram.methods.send_photo, 466
aiogram.methods.send_poll, 469
                                               aiogram.types.background_fill_gradient, 22
aiogram.methods.send_sticker, 373
                                               aiogram.types.background_fill_solid, 22
aiogram.methods.send_venue, 473
                                               aiogram.types.background_type, 23
                                               aiogram.types.background_type_chat_theme, 23
aiogram.methods.send_video, 475
```

714 Python Module Index

```
aiogram.types.background_type_fill, 23
                                               aiogram.types.chat_photo, 131
aiogram.types.background_type_pattern, 24
                                               aiogram.types.chat_shared, 132
                                               aiogram.types.chosen_inline_result, 284
aiogram.types.background_type_wallpaper, 25
aiogram.types.birthdate, 25
                                               aiogram.types.contact, 132
aiogram.types.bot_command, 26
                                               aiogram.types.dice, 133
aiogram.types.bot_command_scope, 26
                                               aiogram.types.document, 133
aiogram.types.bot_command_scope_all_chat_adminaistqpatmortsypes.encrypted_credentials, 349
                                               aiogram.types.encrypted_passport_element, 350
aiogram.types.bot_command_scope_all_group_chataiogram.types.error_event,625
                                               aiogram.types.external_reply_info, 134
aiogram.types.bot_command_scope_all_private_chaitsgram.types.file, 136
                                               aiogram.types.force_reply, 137
aiogram.types.bot_command_scope_chat, 28
                                               aiogram.types.forum_topic, 137
aiogram.types.bot_command_scope_chat_administrantmogram.types.forum_topic_closed, 138
                                               aiogram.types.forum_topic_created, 138
aiogram.types.bot_command_scope_chat_member,
                                               aiogram.types.forum_topic_edited, 139
                                               aiogram.types.forum_topic_reopened, 139
aiogram.types.bot_command_scope_default, 30
                                               aiogram.types.game, 364
aiogram.types.bot_description, 30
                                               aiogram.types.game_high_score, 365
aiogram.types.bot_name, 31
                                               aiogram.types.general_forum_topic_hidden, 139
aiogram.types.bot_short_description, 31
                                               aiogram.types.general_forum_topic_unhidden,
aiogram.types.business_connection, 31
                                                       140
aiogram.types.business_intro, 32
                                               aiogram.types.giveaway, 140
aiogram.types.business_location, 32
                                               aiogram.types.giveaway_completed, 141
aiogram.types.business_messages_deleted, 33
                                               aiogram.types.giveaway_created, 141
aiogram.types.business_opening_hours, 33
                                               aiogram.types.giveaway_winners, 142
aiogram.types.business_opening_hours_interval,aiogram.types.inaccessible_message, 143
                                               aiogram.types.inline_keyboard_button, 183
aiogram.types.callback_game, 364
                                               aiogram.types.inline_keyboard_markup, 184
aiogram.types.callback_query, 34
                                               aiogram.types.inline_query, 284
aiogram.types.chat, 36
                                               aiogram.types.inline_query_result, 286
aiogram.types.chat_administrator_rights, 51
                                               aiogram.types.inline_query_result_article,
aiogram.types.chat_background, 53
aiogram.types.chat_boost,54
                                               aiogram.types.inline_query_result_audio, 288
aiogram.types.chat_boost_added, 54
                                               aiogram.types.inline_query_result_cached_audio,
aiogram.types.chat_boost_removed, 55
aiogram.types.chat_boost_source, 55
                                               aiogram.types.inline_query_result_cached_document,
aiogram.types.chat_boost_source_gift_code,56
aiogram.types.chat_boost_source_giveaway, 56
                                               aiogram.types.inline_query_result_cached_gif,
aiogram.types.chat_boost_source_premium, 57
aiogram.types.chat_boost_updated, 57
                                               aiogram.types.inline_query_result_cached_mpeg4_gif,
aiogram.types.chat_full_info,58
aiogram.types.chat_invite_link,61
                                               aiogram.types.inline_query_result_cached_photo,
aiogram.types.chat_join_request,62
aiogram.types.chat_location, 103
                                               aiogram.types.inline_query_result_cached_sticker,
aiogram.types.chat_member, 103
aiogram.types.chat_member_administrator, 103
                                               aiogram.types.inline_query_result_cached_video,
aiogram.types.chat_member_banned, 106
aiogram.types.chat_member_left, 106
                                               aiogram.types.inline_query_result_cached_voice,
aiogram.types.chat_member_member, 107
aiogram.types.chat_member_owner, 107
                                               aiogram.types.inline_query_result_contact,
aiogram.types.chat_member_restricted, 108
aiogram.types.chat_member_updated, 109
                                               aiogram.types.inline_query_result_document,
aiogram.types.chat_permissions, 130
                                                       308
```

```
aiogram.types.inline_query_result_game, 310
                                               aiogram.types.message_auto_delete_timer_changed,
aiogram.types.inline_query_result_gif, 311
                                                       257
aiogram.types.inline_query_result_location,
                                               aiogram.types.message_entity, 257
                                               aiogram.types.message_id, 258
aiogram.types.inline_query_result_mpeg4_gif,
                                               aiogram.types.message_origin, 258
       315
                                               aiogram.types.message_origin_channel, 259
aiogram.types.inline_query_result_photo, 318
                                               aiogram.types.message_origin_chat, 259
aiogram.types.inline_query_result_venue, 319
                                               aiogram.types.message_origin_hidden_user, 260
aiogram.types.inline_query_result_video, 322
                                               aiogram.types.message_origin_user, 260
aiogram.types.inline_query_result_voice, 324
                                               aiogram.types.message_reaction_count_updated,
aiogram.types.inline_query_results_button,
                                               aiogram.types.message_reaction_updated, 262
aiogram.types.input_contact_message_content,
                                               aiogram.types.order_info,338
        326
                                               aiogram.types.paid_media, 263
aiogram.types.input_file, 184
                                               aiogram.types.paid_media_info, 263
aiogram.types.input_invoice_message_content,
                                               aiogram.types.paid_media_photo, 263
                                               aiogram.types.paid_media_preview, 264
aiogram.types.input_location_message_content,
                                               aiogram.types.paid_media_purchased, 338
                                               aiogram.types.paid_media_video, 264
aiogram.types.input_media, 185
                                               aiogram.types.passport_data, 351
aiogram.types.input_media_animation, 185
                                               aiogram.types.passport_element_error, 352
aiogram.types.input_media_audio, 187
                                               aiogram.types.passport_element_error_data_field,
aiogram.types.input_media_document, 188
aiogram.types.input_media_photo, 189
                                               aiogram.types.passport_element_error_file,
aiogram.types.input_media_video, 190
                                                       353
aiogram.types.input_message_content,330
                                               aiogram.types.passport_element_error_files,
aiogram.types.input_paid_media, 191
aiogram.types.input_paid_media_photo, 192
                                               aiogram.types.passport_element_error_front_side,
aiogram.types.input_paid_media_video, 192
                                                       354
aiogram.types.input_poll_option, 193
                                               aiogram.types.passport_element_error_reverse_side,
aiogram.types.input_sticker, 333
aiogram.types.input_text_message_content,330
                                               aiogram.types.passport_element_error_selfie,
aiogram.types.input_venue_message_content,
                                               aiogram.types.passport_element_error_translation_file,
aiogram.types.invoice, 337
aiogram.types.keyboard_button, 194
                                               aiogram.types.passport_element_error_translation_files,
aiogram.types.keyboard_button_poll_type, 195
aiogram.types.keyboard_button_request_chat,
                                               aiogram.types.passport_element_error_unspecified,
        195
                                                       359
aiogram.types.keyboard_button_request_user,
                                               aiogram.types.passport_file, 360
                                               aiogram.types.photo_size, 265
aiogram.types.keyboard_button_request_users,
                                               aiogram.types.poll, 265
                                               aiogram.types.poll_answer, 266
aiogram.types.labeled_price, 337
                                               aiogram.types.poll_option, 267
aiogram.types.link_preview_options, 199
                                               aiogram.types.pre_checkout_query, 339
aiogram.types.location, 200
                                               aiogram.types.proximity_alert_triggered, 267
aiogram.types.login_url, 200
                                               aiogram.types.reaction_count, 268
aiogram.types.mask_position, 334
                                               aiogram.types.reaction_type, 268
aiogram.types.maybe_inaccessible_message, 201
                                               aiogram.types.reaction_type_custom_emoji, 268
aiogram.types.menu_button, 201
                                               aiogram.types.reaction_type_emoji, 269
aiogram.types.menu_button_commands, 202
                                               aiogram.types.reaction_type_paid, 269
aiogram.types.menu_button_default, 202
                                               aiogram.types.refunded_payment, 340
aiogram.types.menu_button_web_app, 203
                                               aiogram.types.reply_keyboard_markup, 270
                                               aiogram.types.reply_keyboard_remove, 271
aiogram.types.message, 203
```

```
aiogram.types.reply_parameters, 271
aiogram.types.response_parameters, 272
aiogram.types.revenue_withdrawal_state, 340
aiogram.types.revenue_withdrawal_state_failed,
aiogram.types.revenue_withdrawal_state_pending,
aiogram.types.revenue_withdrawal_state_succeeded,
aiogram.types.sent_web_app_message, 333
aiogram.types.shared_user, 273
aiogram.types.shipping_address, 343
aiogram.types.shipping_option, 343
aiogram.types.shipping_query, 344
aiogram.types.star_transaction, 345
aiogram.types.star_transactions, 345
aiogram.types.sticker, 334
aiogram.types.sticker_set,336
aiogram.types.story, 273
aiogram.types.successful_payment, 346
aiogram.types.switch_inline_query_chosen_chat,
aiogram.types.text_quote, 275
aiogram.types.transaction_partner, 347
aiogram.types.transaction_partner_fragment,
aiogram.types.transaction_partner_other, 348
aiogram.types.transaction_partner_telegram_ads,
aiogram.types.transaction_partner_user, 349
aiogram.types.update, 360
aiogram.types.user, 275
aiogram.types.user_chat_boosts, 277
aiogram.types.user_profile_photos, 277
aiogram.types.user_shared, 277
aiogram.types.users_shared, 278
aiogram.types.venue, 279
aiogram.types.video, 279
aiogram.types.video_chat_ended, 280
aiogram.types.video_chat_participants_invited,
aiogram.types.video_chat_scheduled, 281
aiogram.types.video_chat_started, 281
aiogram.types.video_note, 281
aiogram.types.voice, 282
aiogram.types.web_app_data, 283
aiogram.types.web_app_info, 283
aiogram.types.webhook_info, 363
aiogram.types.write_access_allowed, 283
```

## **INDEX**

Symbols method), 630
call() (aiogram.dispatcher.middlewares.base.BaseMiddleware_() (aiogram.webhook.aiohttp_server.TokenBasedRequestHandler method), 623 method), 631
call() (aiogram.filters.base.Filter method), 590
method), 575init() (aiogram.dispatcher.router.Router method), A
accent_color_id (aiogram.types.chat.Chat attribute),
init() (aiogram.filters.command.Command 37 method), 580 accent_color_id(aiogram.types.chat_full_info.ChatFullInfo
init() (aiogram.fsm.storage.memory.MemoryStorage attribute), 58
method), 600 action (aiogram.methods.send_chat_action.SendChatActioninit() (aiogram.fsm.storage.mongo.MongoStorage attribute), 448
method), 601 actions (aiogram.fsm.scene.SceneConfig attribute), 619
init() (aiogram.fsm.storage.redis.RedisStorage active_usernames (aiogram.types.chat.Chat attribute),  method), 600 37
init() (aiogram.types.input_file.BufferedInputFile active_usernames(aiogram.types.chat_full_info.ChatFullInfo attribute), 59
init() (aiogram.types.input_file.FSInputFile actor_chat (aiogram.types.message_reaction_updated.MessageReactionUmethod), 565 attribute), 262
init() (aiogram.utils.callback_answer.CallbackAnswedd() (aiogram.fsm.scene.SceneRegistry method), 617  method), 660  add() (aiogram.utils.keyboard.InlineKeyboardBuilder
init() (aiogram.utils.callback_answer.CallbackAnswerMiddlewnethod), 644
method), 660 add() (aiogram.utils.keyboard.ReplyKeyboardBuilderinit() (aiogram.utils.chat_action.ChatActionSender method), 646
method), 652 add() (aiogram.utils.media_group.MediaGroupBuilder
init() (aiogram.utils.formatting.Text method), 665 method), 669init() (aiogram.utils.i18n.middleware.ConstI18nMidatedwaredio() (aiogram.utils.media_group.MediaGroupBuilder
method), 649 method), 669
init() (aiogram.utils.i18n.middleware.FSMI18nMiddledatate (aiogram.types.chat_boost.ChatBoost at- method), 649 tribute), 54
init() (aiogram.utils.i18n.middleware.I18nMiddlewa <b>re</b> dd_document() (aiogram.utils.media_group.MediaGroupBuilder method), 650 method), 670
init() (aiogram.utils.i18n.middleware.SimpleI18nMi <b>ddl</b> ew <b>pho</b> to() (aiogram.utils.media_group.MediaGroupBuilder method), 648 method), 670
init() (aiogram.utils.keyboard.InlineKeyboardBuildeadd_to_router() (aiogram.fsm.scene.Scene class method), 644 method), 617
init() (aiogram.utils.keyboard.ReplyKeyboardBuilde#dd_video() (aiogram.utils.media_group.MediaGroupBuilder method), 646 method), 671
init() (aiogram.webhook.aiohttp_server.BaseReques#Admdleto_attachment_menu (aiogram.types.user.User method), 630 attribute), 276
init() (aiogram.webhook.aiohttp_server.SimpleRequ <b>edtled.dto</b> _attachment_menu

	(aiogram.utils.web_app.WebAppUser at-tribute), 657	<pre>aiogram.enums.keyboard_button_poll_type_typ module,556</pre>
addition	nal_chat_count (aiogram.types.giveaway_winners.GiveawayWinn	aiogram.enums.mask_position_point
	attribute), 142	aiogram.enums.menu_button_type
	(aiogram.enums.encrypted_passport_element.Enc	
	attribute), 554	aiogram.enums.message_entity_type
	(aiogram.methods.send_venue.SendVenue at-	module, 557
	tribute), 473	aiogram.enums.message_origin_type
	(aiogram.types.business_location.BusinessLocati	
	attribute), 32	aiogram.enums.paid_media_type
	(aiogram.types.chat_location.ChatLocation at-	module, 558
	tribute), 103	aiogram.enums.parse_mode
	(aiogram.types.inline_query_result_venue.InlineQ	
	attribute), 320	aiogram.enums.passport_element_error_type
	(aiogram.types.input_venue_message_content.Inp	
	attribute), 332	aiogram.enums.poll_type
	(aiogram.types.venue.Venue attribute), 279	module, 559
	kerToSet (class in	
	aiogram.methods.add_sticker_to_set), 365	module, 560
adjust()		aiogram.enums.revenue_withdrawal_state_type module, 560
adjust()	) (aiogram.utils.keyboard.ReplyKeyboardBuilder	aiogram.enums.sticker_format
	method), 646	module, 560
ADMINIST	TRATOR (aiogram.enums.chat_member_status.Cha	utalilengbamSternums.sticker_type
	attribute), 548	module, 561
AED (aiog	gram.enums.currency.Currency attribute), 551	aiogram.enums.topic_icon_color
AFN (aiog	gram.enums.currency.Currency attribute), 551	module, 561
	.dispatcher.flags	aiogram.enums.transaction_partner_type
modu	ıle, 628	module, 561
aiogram	.enums.bot_command_scope_type	aiogram.enums.update_type
modu	ıle, 547	module, 562
aiogram	.enums.chat_action	aiogram.exceptions
	ıle, 547	module, 626
aiogram	.enums.chat_boost_source_type	aiogram.handlers.callback_query
modu	ale, 548	module, 639
_	.enums.chat_member_status ule,548	<pre>aiogram.methods.add_sticker_to_set   module, 365</pre>
	.enums.chat_type ale,549	<pre>aiogram.methods.answer_callback_query   module, 384</pre>
_	.enums.content_type ale,549	<pre>aiogram.methods.answer_inline_query module, 521</pre>
	.enums.currency ule,551	<pre>aiogram.methods.answer_pre_checkout_query   module,530</pre>
_	.enums.dice_emoji ule,554	<pre>aiogram.methods.answer_shipping_query   module,531</pre>
_	.enums.encrypted_passport_element ule,554	<pre>aiogram.methods.answer_web_app_query   module, 524</pre>
_	.enums.inline_query_result_type ule,555	<pre>aiogram.methods.approve_chat_join_request   module, 385</pre>
	.enums.input_media_type	aiogram.methods.ban_chat_member
	ale, 556	module, 386
aiogram	.enums.input_paid_media_type	aiogram.methods.ban_chat_sender_chat module,388

aiogram.methods.close module, 389	<pre>aiogram.methods.edit_message_reply_markup module,515</pre>
aiogram.methods.close_forum_topic module,390	aiogram.methods.edit_message_text module, 516
aiogram.methods.close_general_forum_topic	aiogram.methods.export_chat_invite_link
module, 391	module, 411
aiogram.methods.copy_message	aiogram.methods.forward_message
module, 392	module, 412
aiogram.methods.copy_messages module, 394	aiogram.methods.forward_messages module,414
aiogram.methods.create_chat_invite_link module, 396	aiogram.methods.get_business_connection module, 415
<pre>aiogram.methods.create_chat_subscription_invi module, 398</pre>	tæidgiræm.methods.get_chat module,416
aiogram.methods.create_forum_topic	aiogram.methods.get_chat_administrators
module, 399	module, 417
<pre>aiogram.methods.create_invoice_link   module, 533</pre>	aiogram.methods.get_chat_member module,418
<pre>aiogram.methods.create_new_sticker_set   module, 366</pre>	<pre>aiogram.methods.get_chat_member_count module,419</pre>
${\tt aiogram.methods.decline\_chat\_join\_request} \\ {\tt module,} \ 400$	aiogram.methods.get_chat_menu_button module,420
aiogram.methods.delete_chat_photo	aiogram.methods.get_custom_emoji_stickers
module, 402	module, 370
aiogram.methods.delete_chat_sticker_set	aiogram.methods.get_file
module, 403	module, 421
aiogram.methods.delete_forum_topic module, 404	<pre>aiogram.methods.get_forum_topic_icon_stickers module, 422</pre>
<pre>aiogram.methods.delete_message   module, 507</pre>	<pre>aiogram.methods.get_game_high_scores module, 525</pre>
aiogram.methods.delete_messages module, 508	<pre>aiogram.methods.get_me module, 423</pre>
aiogram.methods.delete_my_commands module, 405	<pre>aiogram.methods.get_my_commands   module, 424</pre>
aiogram.methods.delete_sticker_from_set module, 368	aiogram.methods.get_my_default_administrator_rights module, 425
aiogram.methods.delete_sticker_set module, 369	aiogram.methods.get_my_description module, 426
aiogram.methods.delete_webhook module, 540	aiogram.methods.get_my_name module, 427
aiogram.methods.edit_chat_invite_link module, 406	aiogram.methods.get_my_short_description module, 428
aiogram.methods.edit_chat_subscription_invite module, 408	
aiogram.methods.edit_forum_topic module, 409	aiogram.methods.get_sticker_set module, 371
aiogram.methods.edit_general_forum_topic module, 410	aiogram.methods.get_updates module,541
aiogram.methods.edit_message_caption	aiogram.methods.get_user_chat_boosts
module, 509	module, 428
<pre>aiogram.methods.edit_message_live_location    module, 511</pre>	<pre>aiogram.methods.get_user_profile_photos   module, 429</pre>
aiogram.methods.edit_message_media	aiogram.methods.get_webhook_info
module, 513	module, 543

<pre>aiogram.methods.hide_general_forum_topic    module, 430</pre>	aiogram.methods.send_video module,475
<pre>aiogram.methods.leave_chat   module, 431</pre>	<pre>aiogram.methods.send_video_note   module, 479</pre>
aiogram.methods.log_out module,432	aiogram.methods.send_voice module,481
aiogram.methods.pin_chat_message module, 433	aiogram.methods.set_chat_administrator_custom_title module, 484
aiogram.methods.promote_chat_member module, 434	aiogram.methods.set_chat_description module, 485
aiogram.methods.refund_star_payment module, 536	aiogram.methods.set_chat_menu_button module,486
aiogram.methods.reopen_forum_topic module,437	aiogram.methods.set_chat_permissions module, 487
aiogram.methods.reopen_general_forum_topic module,438	aiogram.methods.set_chat_photo module, 489
aiogram.methods.replace_sticker_in_set module, 372	aiogram.methods.set_chat_sticker_set module, 490
<pre>aiogram.methods.restrict_chat_member module, 439</pre>	aiogram.methods.set_chat_title module,491
aiogram.methods.revoke_chat_invite_link module, 440	<pre>aiogram.methods.set_custom_emoji_sticker_set_thumbnail    module, 375</pre>
aiogram.methods.send_animation module, 442	<pre>aiogram.methods.set_game_score   module, 529</pre>
aiogram.methods.send_audio module,445	aiogram.methods.set_message_reaction module,492
<pre>aiogram.methods.send_chat_action   module, 447</pre>	<pre>aiogram.methods.set_my_commands   module, 493</pre>
<pre>aiogram.methods.send_contact   module, 449</pre>	<pre>aiogram.methods.set_my_default_administrator_rights   module, 495</pre>
<pre>aiogram.methods.send_dice   module, 451</pre>	<pre>aiogram.methods.set_my_description   module, 496</pre>
<pre>aiogram.methods.send_document   module, 453</pre>	<pre>aiogram.methods.set_my_name   module, 497</pre>
aiogram.methods.send_game module,526	<pre>aiogram.methods.set_my_short_description   module, 498</pre>
aiogram.methods.send_invoice module,537	<pre>aiogram.methods.set_passport_data_errors module, 545</pre>
aiogram.methods.send_location module, 456	<pre>aiogram.methods.set_sticker_emoji_list   module, 376</pre>
<pre>aiogram.methods.send_media_group   module, 459</pre>	<pre>aiogram.methods.set_sticker_keywords   module, 377</pre>
aiogram.methods.send_message module, 461	<pre>aiogram.methods.set_sticker_mask_position   module, 378</pre>
aiogram.methods.send_paid_media module,464	<pre>aiogram.methods.set_sticker_position_in_set   module, 379</pre>
aiogram.methods.send_photo module,466	<pre>aiogram.methods.set_sticker_set_thumbnail   module, 380</pre>
aiogram.methods.send_poll module,469	<pre>aiogram.methods.set_sticker_set_title   module, 382</pre>
aiogram.methods.send_sticker module,373	<pre>aiogram.methods.set_webhook   module, 543</pre>
aiogram.methods.send_venue	aiogram.methods.stop_message_live_location

```
aiogram.methods.stop_poll
                                                aiogram.types.bot_command_scope_chat_administrators
   module, 520
                                                    module, 29
aiogram.methods.unban_chat_member
                                                aiogram.types.bot_command_scope_chat_member
   module, 499
                                                    module, 29
aiogram.methods.unban_chat_sender_chat
                                                aiogram.types.bot_command_scope_default
   module, 500
                                                    module, 30
aiogram.methods.unhide_general_forum_topic
                                                aiogram.types.bot_description
    module, 501
                                                    module, 30
aiogram.methods.unpin_all_chat_messages
                                                aiogram.types.bot_name
    module, 502
                                                    module, 31
aiogram.methods.unpin_all_forum_topic_messagesaiogram.types.bot_short_description
    module, 503
                                                    module, 31
aiogram.methods.unpin_all_general_forum_topic_amesssamestypes.business_connection
    module, 504
                                                    module, 31
aiogram.methods.unpin_chat_message
                                                aiogram.types.business_intro
    module, 505
                                                    module, 32
aiogram.methods.upload_sticker_file
                                                aiogram.types.business_location
    module, 383
                                                    module, 32
aiogram.types.animation
                                                aiogram.types.business_messages_deleted
    module, 19
                                                    module, 33
aiogram.types.audio
                                                aiogram.types.business_opening_hours
    module, 20
                                                    module, 33
aiogram.types.background_fill
                                                aiogram.types.business_opening_hours_interval
    module, 21
                                                    module, 34
aiogram.types.background_fill_freeform_gradienatiogram.types.callback_game
    module, 21
                                                    module, 364
aiogram.types.background_fill_gradient
                                                aiogram.types.callback_query
    module, 22
                                                    module, 34
aiogram.types.background_fill_solid
                                                aiogram.types.chat
    module, 22
                                                    module, 36
aiogram.types.background_type
                                                aiogram.types.chat_administrator_rights
    module, 23
                                                    module, 51
aiogram.types.background_type_chat_theme
                                                aiogram.types.chat_background
    module, 23
                                                    module, 53
aiogram.types.background_type_fill
                                                aiogram.types.chat_boost
                                                    module, 54
   module, 23
aiogram.types.background_type_pattern
                                                aiogram.types.chat_boost_added
    module, 24
                                                    module, 54
aiogram.types.background_type_wallpaper
                                                aiogram.types.chat_boost_removed
                                                    module, 55
   module, 25
aiogram.types.birthdate
                                                aiogram.types.chat_boost_source
    module, 25
                                                    module, 55
aiogram.types.bot_command
                                                aiogram.types.chat_boost_source_gift_code
    module, 26
                                                    module, 56
aiogram.types.bot_command_scope
                                                aiogram.types.chat_boost_source_giveaway
    module, 26
                                                    module, 56
aiogram.types.bot_command_scope_all_chat_adminaisdgradmortsypes.chat_boost_source_premium
    module, 27
                                                    module, 57
aiogram.types.bot_command_scope_all_group_chatasiogram.types.chat_boost_updated
    module, 27
                                                    module, 57
aiogram.types.bot_command_scope_all_private_chamicsgram.types.chat_full_info
    module, 28
                                                    module, 58
aiogram.types.bot_command_scope_chat
                                                aiogram.types.chat_invite_link
    module, 28
                                                    module, 61
```

<pre>aiogram.types.chat_join_request   module,62</pre>	<pre>aiogram.types.forum_topic_reopened   module, 139</pre>
<pre>aiogram.types.chat_location   module, 103</pre>	aiogram.types.game module,364
<pre>aiogram.types.chat_member module, 103</pre>	<pre>aiogram.types.game_high_score   module, 365</pre>
<pre>aiogram.types.chat_member_administrator   module, 103</pre>	<pre>aiogram.types.general_forum_topic_hidden   module, 139</pre>
<pre>aiogram.types.chat_member_banned   module, 106</pre>	aiogram.types.general_forum_topic_unhidden module,140
<pre>aiogram.types.chat_member_left   module, 106</pre>	aiogram.types.giveaway module,140
<pre>aiogram.types.chat_member_member module, 107</pre>	aiogram.types.giveaway_completed module,141
<pre>aiogram.types.chat_member_owner   module, 107</pre>	<pre>aiogram.types.giveaway_created   module, 141</pre>
aiogram.types.chat_member_restricted module, 108	aiogram.types.giveaway_winners module,142
<pre>aiogram.types.chat_member_updated   module, 109</pre>	aiogram.types.inaccessible_message module,143
<pre>aiogram.types.chat_permissions   module, 130</pre>	<pre>aiogram.types.inline_keyboard_button   module, 183</pre>
<pre>aiogram.types.chat_photo   module, 131</pre>	aiogram.types.inline_keyboard_markup module,184
<pre>aiogram.types.chat_shared   module, 132</pre>	aiogram.types.inline_query module,284
aiogram.types.chosen_inline_result module, 284	aiogram.types.inline_query_result module,286
aiogram.types.contact module, 132	aiogram.types.inline_query_result_article module,287
aiogram.types.dice module, 133	aiogram.types.inline_query_result_audio module,288
aiogram.types.document module, 133	aiogram.types.inline_query_result_cached_audio module,290
aiogram.types.encrypted_credentials module, 349	<pre>aiogram.types.inline_query_result_cached_document    module, 292</pre>
<pre>aiogram.types.encrypted_passport_element    module, 350</pre>	<pre>aiogram.types.inline_query_result_cached_gif   module, 294</pre>
aiogram.types.error_event module,625	<pre>aiogram.types.inline_query_result_cached_mpeg4_gif module, 296</pre>
<pre>aiogram.types.external_reply_info   module, 134</pre>	<pre>aiogram.types.inline_query_result_cached_photo   module, 298</pre>
aiogram.types.file module, 136	<pre>aiogram.types.inline_query_result_cached_sticker module, 300</pre>
<pre>aiogram.types.force_reply   module, 137</pre>	<pre>aiogram.types.inline_query_result_cached_video   module, 302</pre>
<pre>aiogram.types.forum_topic   module,137</pre>	<pre>aiogram.types.inline_query_result_cached_voice   module, 304</pre>
<pre>aiogram.types.forum_topic_closed   module, 138</pre>	<pre>aiogram.types.inline_query_result_contact   module, 306</pre>
aiogram.types.forum_topic_created module, 138	<pre>aiogram.types.inline_query_result_document    module, 308</pre>
aiogram.types.forum_topic_edited	aiogram.types.inline_query_result_game module.310

<pre>aiogram.types.inline_query_result_gif   module, 311</pre>	aiogram.types.keyboard_button module,194
<pre>aiogram.types.inline_query_result_location    module, 313</pre>	<pre>aiogram.types.keyboard_button_poll_type   module, 195</pre>
<pre>aiogram.types.inline_query_result_mpeg4_gif   module, 315</pre>	<pre>aiogram.types.keyboard_button_request_chat   module, 195</pre>
<pre>aiogram.types.inline_query_result_photo   module, 318</pre>	<pre>aiogram.types.keyboard_button_request_user   module, 197</pre>
<pre>aiogram.types.inline_query_result_venue   module, 319</pre>	<pre>aiogram.types.keyboard_button_request_users   module, 197</pre>
<pre>aiogram.types.inline_query_result_video   module, 322</pre>	<pre>aiogram.types.labeled_price   module, 337</pre>
<pre>aiogram.types.inline_query_result_voice   module, 324</pre>	<pre>aiogram.types.link_preview_options   module, 199</pre>
<pre>aiogram.types.inline_query_results_button   module, 325</pre>	aiogram.types.location module, 200
<pre>aiogram.types.input_contact_message_content    module, 326</pre>	aiogram.types.login_url module,200
<pre>aiogram.types.input_file   module, 184</pre>	<pre>aiogram.types.mask_position   module, 334</pre>
<pre>aiogram.types.input_invoice_message_content   module, 326</pre>	<pre>aiogram.types.maybe_inaccessible_message   module, 201</pre>
<pre>aiogram.types.input_location_message_content    module, 329</pre>	<pre>aiogram.types.menu_button   module, 201</pre>
<pre>aiogram.types.input_media   module, 185</pre>	aiogram.types.menu_button_commands module, 202
<pre>aiogram.types.input_media_animation   module, 185</pre>	aiogram.types.menu_button_default module, 202
<pre>aiogram.types.input_media_audio   module, 187</pre>	<pre>aiogram.types.menu_button_web_app   module, 203</pre>
aiogram.types.input_media_document module, 188	aiogram.types.message module, 203
<pre>aiogram.types.input_media_photo   module, 189</pre>	aiogram.types.message_auto_delete_timer_changed module, 257
aiogram.types.input_media_video module, 190	aiogram.types.message_entity module, 257
<pre>aiogram.types.input_message_content   module, 330</pre>	aiogram.types.message_id module, 258
<pre>aiogram.types.input_paid_media   module, 191</pre>	aiogram.types.message_origin module,258
<pre>aiogram.types.input_paid_media_photo   module, 192</pre>	<pre>aiogram.types.message_origin_channel    module, 259</pre>
<pre>aiogram.types.input_paid_media_video   module, 192</pre>	<pre>aiogram.types.message_origin_chat   module, 259</pre>
<pre>aiogram.types.input_poll_option   module, 193</pre>	aiogram.types.message_origin_hidden_user module, 260
<pre>aiogram.types.input_sticker module, 333</pre>	aiogram.types.message_origin_user module,260
<pre>aiogram.types.input_text_message_content    module, 330</pre>	<pre>aiogram.types.message_reaction_count_updated   module, 261</pre>
<pre>aiogram.types.input_venue_message_content    module, 331</pre>	<pre>aiogram.types.message_reaction_updated   module, 262</pre>
aiogram.types.invoice module, 337	<pre>aiogram.types.order_info   module, 338</pre>

aiogram.types.paid_media module,263	<pre>aiogram.types.reaction_type_emoji   module, 269</pre>
aiogram.types.paid_media_info module, 263	<pre>aiogram.types.reaction_type_paid   module, 269</pre>
aiogram.types.paid_media_photo module,263	<pre>aiogram.types.refunded_payment   module, 340</pre>
aiogram.types.paid_media_preview module,264	aiogram.types.reply_keyboard_markup module,270
aiogram.types.paid_media_purchased module,338	<pre>aiogram.types.reply_keyboard_remove   module, 271</pre>
aiogram.types.paid_media_video module,264	aiogram.types.reply_parameters module,271
aiogram.types.passport_data module,351	<pre>aiogram.types.response_parameters   module, 272</pre>
aiogram.types.passport_element_error module, 352	aiogram.types.revenue_withdrawal_state module, 340
module, 352	<pre>latiogram.types.revenue_withdrawal_state_failed    module, 341</pre>
<pre>aiogram.types.passport_element_error_file   module, 353</pre>	<pre>aiogram.types.revenue_withdrawal_state_pending   module, 341</pre>
aiogram.types.passport_element_error_files module,354	<pre>aiogram.types.revenue_withdrawal_state_succeeded   module, 342</pre>
<pre>aiogram.types.passport_element_error_front_si   module, 354</pre>	module, 333
<pre>aiogram.types.passport_element_error_reverse_     module, 355</pre>	module, 273
aiogram.types.passport_element_error_selfie module,356	aiogram.types.shipping_address module, 343
<pre>aiogram.types.passport_element_error_translat   module, 357</pre>	module, 343
<pre>aiogram.types.passport_element_error_translat   module, 358</pre>	module, 344
<pre>aiogram.types.passport_element_error_unspecif module, 359</pre>	<pre>iæribgram.types.star_transaction   module, 345</pre>
aiogram.types.passport_file module,360	<pre>aiogram.types.star_transactions   module, 345</pre>
aiogram.types.photo_size module,265	aiogram.types.sticker module,334
aiogram.types.poll module, 265	aiogram.types.sticker_set module,336
aiogram.types.poll_answer module,266	aiogram.types.story module,273
aiogram.types.poll_option module,267	<pre>aiogram.types.successful_payment   module, 346</pre>
<pre>aiogram.types.pre_checkout_query   module, 339</pre>	<pre>aiogram.types.switch_inline_query_chosen_chat   module, 274</pre>
aiogram.types.proximity_alert_triggered module, 267	<pre>aiogram.types.text_quote   module, 275</pre>
aiogram.types.reaction_count module,268	aiogram.types.transaction_partner module,347
aiogram.types.reaction_type module,268	<pre>aiogram.types.transaction_partner_fragment   module, 347</pre>
aiogram.types.reaction_type_custom_emoji module, 268	aiogram.types.transaction_partner_other module, 348

<pre>aiogram.types.transaction_partner_telegram_ad   module, 348</pre>	s (aiogram.types.switch_inline_query_chosen_chat.SwitchInlineQuattribute), 274
<pre>aiogram.types.transaction_partner_user   module, 349</pre>	allow_group_chats (aiogram.types.switch_inline_query_chosen_chat.Sw attribute), 274
aiogram.types.update	allow_sending_without_reply
module, 360	(aiogram.client.default.DefaultBotProperties
aiogram.types.user	attribute), 567
module, 275	allow_sending_without_reply
aiogram.types.user_chat_boosts	(aiogram.methods.copy_message.CopyMessage
module, 277	attribute), 393
aiogram.types.user_profile_photos	allow_sending_without_reply
module, 277	(aiogram.methods.send_animation.SendAnimation
aiogram.types.user_shared	attribute), 443
module, 277	allow_sending_without_reply
aiogram.types.users_shared	(aiogram.methods.send_audio.SendAudio
module, 278	attribute), 446
aiogram.types.venue	allow_sending_without_reply
module, 279	(aiogram.methods.send_contact.SendContact
aiogram.types.video	attribute), 450
module, 279	allow_sending_without_reply
aiogram.types.video_chat_ended	(aiogram.methods.send_dice.SendDice at-
module, 280	tribute), 452
aiogram.types.video_chat_participants_invited	· · · · · · · · · · · · · · · · · · ·
module, 280	(aiogram.methods.send_document.SendDocument
	attribute), 455
aiogram.types.video_chat_scheduled module, 281	allow_sending_without_reply
aiogram.types.video_chat_started	(aiogram.methods.send_game.SendGame
module, 281	attribute), 527
aiogram.types.video_note	allow_sending_without_reply
module, 281	(aiogram.methods.send_invoice.SendInvoice
aiogram.types.voice	attribute), 539
module, 282	allow_sending_without_reply
aiogram.types.web_app_data	(aiogram.methods.send_location.SendLocation
module, 283	attribute), 458
aiogram.types.web_app_info	allow_sending_without_reply
module, 283	(aiogram.methods.send_media_group.SendMediaGroup
aiogram.types.webhook_info	attribute), 460
module, 363	allow_sending_without_reply
aiogram.types.write_access_allowed	(aiogram.methods.send_message.SendMessage
module, 283	attribute), 463
AiogramError, 626	allow_sending_without_reply
AiohttpSession (class in	(aiogram.methods.send_photo.SendPhoto
aiogram.client.session.aiohttp), 16	attribute), 468
ALL (aiogram.enums.currency.Currency attribute), 551	allow_sending_without_reply
ALL_CHAT_ADMINISTRATORS	(aiogram.methods.send_poll.SendPoll at-
(aiogram.enums.bot_command_scope_type.BotC	1 11.
attribute), 547	allow_sending_without_reply
${\tt ALL\_GROUP\_CHATS}\ (a iogram.enums.bot\_command\_scope\_$	
attribute), 547	attribute), 374
${\tt ALL\_PRIVATE\_CHATS} \ (a iogram.enums.bot\_command\_scoperation) \\$	7.5
attribute), 547	(aiogram.methods.send_venue.SendVenue
$\verb allow_bot_chats  (a iogram. types. switch_in line\_query\_charmonter)                                      $	
attribute), 274	allow_sending_without_reply
allow_channel_chats	(aiogram.methods.send_video.SendVideo

answer()

method), 285

```
attribute), 477
                                                                                                                   answer() (aiogram.types.message.Message method),
allow_sending_without_reply
                   (aiogram.methods.send_video_note.SendVideoNotanswer() (aiogram.types.pre_checkout_query.PreCheckoutQuery
                   attribute), 480
                                                                                                                                       method), 339
allow_sending_without_reply
                                                                                                                    answer() (aiogram.types.shipping_query.ShippingQuery
                   (aiogram.methods.send\_voice.SendVoice
                                                                                                                                       method), 344
                   attribute), 482
                                                                                                                    answer_animation() (aiogram.types.chat_join_request.ChatJoinRequest
allow_sending_without_reply
                                                                                                                                       method), 65
                   (aiogram.types.reply_parameters.ReplyParameteranswer_animation() (aiogram.types.chat_member_updated.ChatMembe
                   attribute), 271
                                                                                                                                       method), 111
allow_user_chats(aiogram.types.switch_inline_query_chanswerhansswerhansswerhansee.huatechats(aiogram.types.switch_inline_query_chansswerhansee.huatechats(aiogram.types.switch_inline_query_chansswerhansee.huatechats)
                   attribute), 274
                                                                                                                                       method), 145
allowed_updates(aiogram.methods.get_updates.GetUpdatesswer_animation() (aiogram.types.message.Message
                  attribute), 542
                                                                                                                                       method), 212
allowed_updates(aiogram.methods.set_webhook.SetWebhamsbwer_animation_pm()
                   attribute), 544
                                                                                                                                        (aiogram.types.chat_join_request.ChatJoinRequest
allowed_updates (aiogram.types.webhook_info.WebhookInfo
                                                                                                                                       method), 67
                   attribute), 363
                                                                                                                    answer_audio() (aiogram.types.chat_join_request.ChatJoinRequest
allows_multiple_answers
                                                                                                                                       method), 68
                                                                                                                   answer_audio() (aiogram.types.chat_member_updated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberU
                   (aiogram.methods.send_poll.SendPoll
                   tribute), 471
                                                                                                                                       method), 112
allows_multiple_answers
                                                              (aiogram.types.poll.Poll
                                                                                                                   answer_audio() (aiogram.types.inaccessible_message.InaccessibleMessa
                   attribute), 266
                                                                                                                                       method), 148
allows_write_to_pm(aiogram.utils.web_app.WebAppUsanswer_audio()
                                                                                                                                                                  (aiogram.types.message.Message
                                                                                                                                       method), 214
                   attribute), 657
AMD (aiogram.enums.currency.Currency attribute), 551
                                                                                                                    answer_audio_pm() (aiogram.types.chat_join_request.ChatJoinRequest
amount \ (aiogram.types.labeled\_price.LabeledPrice \ at-
                                                                                                                                       method), 70
                   tribute), 337
                                                                                                                   answer_contact() (aiogram.types.chat_join_request.ChatJoinRequest
amount (aiogram.types.star_transaction.StarTransaction
                                                                                                                                       method), 71
                   attribute), 345
                                                                                                                    answer_contact() (aiogram.types.chat_member_updated.ChatMemberU
ANIMATED (aiogram.enums.sticker_format.StickerFormat
                                                                                                                                       method), 114
                   attribute), 560
                                                                                                                   answer_contact() (aiogram.types.inaccessible_message.InaccessibleMes
ANIMATION
                          (aiogram.enums.content_type.ContentType
                                                                                                                                       method), 150
                   attribute), 549
                                                                                                                    answer_contact()
                                                                                                                                                                  (aiogram.types.message.Message
ANIMATION (aiogram.enums.input_media_type.InputMediaType
                                                                                                                                       method), 216
                  attribute), 556
                                                                                                                    answer_contact_pm()
\verb"animation" (a iogram.methods.send\_animation.SendAnimation")
                                                                                                                                       (aiogram.types.chat_join_request.ChatJoinRequest
                   attribute), 442
                                                                                                                                       method), 72
animation(aiogram.types.external_reply_info.ExternalReplybyfer_dice() (aiogram.types.chat_join_request.ChatJoinRequest
                   attribute), 135
                                                                                                                                       method), 90
animation (aiogram.types.game.Game attribute), 364
                                                                                                                   answer_dice() (aiogram.types.chat member updated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated
animation (aiogram.types.message.Message attribute),
                                                                                                                                       method), 123
                                                                                                                    answer_dice() (aiogram.types.inaccessible_message.InaccessibleMessag
Animation (class in aiogram.types.animation), 19
                                                                                                                                       method), 168
answer() (aiogram.types.callback_query.CallbackQuery
                                                                                                                 answer_dice()
                                                                                                                                                                  (aiogram.types.message.Message
                   method), 35
                                                                                                                                        method), 235
answer() (aiogram.types.chat_join_request.ChatJoinRequeatnswer_dice_pm() (aiogram.types.chat_join_request.ChatJoinRequest
                                                                                                                                       method), 90
                   method), 63
answer() (aiogram.types.chat_member_updated.ChatMembarskybedtytedcument() (aiogram.types.chat_join_request.ChatJoinRequest
                   method), 110
                                                                                                                                        method), 73
answer() (aiogram.types.inaccessible_message.InaccessibleatMessagelocument() (aiogram.types.chat_member_updated.ChatMember
                  method), 143
                                                                                                                                       method), 115
```

728 Index

(aiogram.types.inline\_query.InlineQuery answer\_document() (aiogram.types.inaccessible\_message.InaccessibleMo

method), 152

(aiogram.types.message.Message

```
answer_document() (aiogram.types.message.Message
                                                                                                                                                                                                           method), 181
                            method), 218
                                                                                                                                                                              answer_paid_media()
                                                                                                                                                                                                           (aiogram.types.message.Message
answer_document_pm()
                                                                                                                                                                                                                                                                                                                            method),
                             (aiogram.types.chat_join_request.ChatJoinRequest
                            method), 74
                                                                                                                                                                              answer_photo() (aiogram.types.chat_join_request.ChatJoinRequest
answer_game() (aiogram.types.chat join request.ChatJoinRequest method), 84
                                                                                                                                                                              answer_photo() (aiogram.types.chat member updated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdate
                             method), 75
answer_game() (aiogram.types.chat_member_updated.ChatMemberbhedlaned), 120
                             method), 116
                                                                                                                                                                              answer_photo() (aiogram.types.inaccessible_message.InaccessibleMessa
answer_game() (aiogram.types.inaccessible_message.InaccessibleMmsethged), 163
                            method), 154
                                                                                                                                                                              answer_photo()
                                                                                                                                                                                                                                                    (aiogram.types.message.Message
answer_game()
                                                                                                                                                                                                           method), 231
                                                                      (aiogram.types.message.Message
                            method), 220
                                                                                                                                                                              answer_photo_pm() (aiogram.types.chat_join_request.ChatJoinRequest
answer_game_pm() (aiogram.types.chat_join_request.ChatJoinRequesethod), 85
                                                                                                                                                                              answer_pm() (aiogram.types.chat_join_request.ChatJoinRequest
                             method), 76
answer_invoice() (aiogram.types.chat_join_request.ChatJoinRequestthod), 64
                                                                                                                                                                              answer_poll() (aiogram.types.chat_join_request.ChatJoinRequest
                             method), 77
answer_invoice() (aiogram.types.chat_member_updated.ChatMembrethindiated
                                                                                                                                                                              answer_poll() (aiogram.types.chat_member_updated.ChatMemberUpdated.
                            method), 117
answer_invoice() (aiogram.types.inaccessible_message.InaccessibleMessage22
                            method), 156
                                                                                                                                                                              answer_poll() (aiogram.types.inaccessible_message.InaccessibleMessag
answer_invoice()
                                                                      (aiogram.types.message.Message
                                                                                                                                                                                                           method), 165
                             method), 223
                                                                                                                                                                              answer_poll()
                                                                                                                                                                                                                                                    (aiogram.types.message.Message
answer_invoice_pm()
                                                                                                                                                                                                           method), 233
                             (aiogram.types.chat_join_request.ChatJoinRequesanswer_poll_pm() (aiogram.types.chat_join_request.ChatJoinRequesa
                            method), 78
                                                                                                                                                                                                           method), 88
answer_location() (aiogram.types.chat_join_request.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowereq
                                                                                                                                                                                                           method), 91
                            method), 80
answer_location() (aiogram.types.chat_member_updatedxCswatMestbeoklepdQtediogram.types.chat_member_updated.ChatMemberU
                             method), 118
                                                                                                                                                                                                           method), 124
answer_location() (aiogram.types.inaccessible_messagealmewers.siblivelkess@e(aiogram.types.inaccessible_message.InaccessibleMes
                             method), 159
                                                                                                                                                                                                           method), 170
answer_location() (aiogram.types.message.Message
                                                                                                                                                                              answer_sticker()
                                                                                                                                                                                                                                                   (aiogram.types.message.Message
                             method), 226
                                                                                                                                                                                                           method), 237
answer_location_pm()
                                                                                                                                                                              answer_sticker_pm()
                             (aiogram.types.chat_join_request.ChatJoinRequest
                                                                                                                                                                                                           (aiogram.types.chat_join_request.ChatJoinRequest
                            method), 82
                                                                                                                                                                                                            method), 92
answer_media_group()
                                                                                                                                                                              answer_venue() (aiogram.types.chat_join_request.ChatJoinRequest
                             (aiogram.types.chat_join_request.ChatJoinRequest
                                                                                                                                                                                                           method), 93
                            method), 83
                                                                                                                                                                              answer_venue() (aiogram.types.chat_member_updated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdate
answer_media_group()
                                                                                                                                                                                                           method), 125
                             (aiogram.types.chat_member_updated.ChatMembanlspdatedenue() (aiogram.types.inaccessible_message.InaccessibleMessa
                            method), 120
                                                                                                                                                                                                           method), 171
answer_media_group()
                                                                                                                                                                              answer_venue()
                                                                                                                                                                                                                                                     (aiogram.types.message.Message
                             (aiogram.types.inaccessible_message.InaccessibleMessage method), 239
                            method), 162
                                                                                                                                                                              answer_venue_pm() (aiogram.types.chat_join_request.ChatJoinRequest
answer_media_group()
                                                                                                                                                                                                           method), 94
                                                                                                                                             method),
                                                                                                                                                                              answer_video() (aiogram.types.chat_join_request.ChatJoinRequest
                             (aiogram.types.message.Message
                             227
                                                                                                                                                                                                           method), 95
                                                                                                                                                                              answer_video() (aiogram.types.chat_member_updated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberUpdated.ChatMemberU
answer_media_group_pm()
                             (aiogram.types.chat_join_request.ChatJoinRequest
                                                                                                                                                                                                           method), 126
                                                                                                                                                                              answer_video() (aiogram.types.inaccessible_message.InaccessibleMessa
                            method), 83
answer_paid_media()
                                                                                                                                                                                                           method), 173
```

Index 729

(aiogram.types.inaccessible message.Inaccessibleatdswergevideo()

<pre>method), 241 answer_video_note()</pre>	args (aiogram.filters.command.CommandObject attribute), 580
	esARS (aiogram.enums.currency.Currency attribute), 551
method), 98	ARTICLE (aiogram.enums.inline_query_result_type.InlineQueryResultTyp
<pre>answer_video_note()</pre>	attribute), 555
(aiogram.types.chat_member_updated.ChatMem	b <b>asUpdatHer()</b> (aiogram.fsm.scene.Scene class method),
method), 127	617
<pre>answer_video_note()</pre>	as_html() (aiogram.utils.formatting.Text method), 665
(aiogram.types.inaccessible_message.Inaccessible method), 176	lealstestsergevalue() (in module aiogram.utils.formatting), 663
<pre>answer_video_note()</pre>	as_kwargs() (aiogram.utils.formatting.Text method),
(aiogram.types.message.Message method),	665
243	as_line() (in module aiogram.utils.formatting), 662
<pre>answer_video_note_pm()</pre>	as_list() (in module aiogram.utils.formatting), 662
(aiogram.types.chat_join_request.ChatJoinReque method), 99	esats_markdown() (aiogram.utils.formatting.Text method), 665
answer_video_pm() (aiogram.types.chat_join_request.Cl	
method), 97	aiogram.utils.formatting), 662
answer_voice() (aiogram.types.chat_join_request.ChatJ	
method), 100	aiogram.utils.formatting), 663
<pre>answer_voice() (aiogram.types.chat_member_updated.C</pre>	
method), 128	aiogram.utils.formatting), 662
<pre>answer_voice() (aiogram.types.inaccessible_message.In</pre>	aasemindeAredasection() (in module
method), 178	aiogram.utils.formatting), 663
<pre>answer_voice() (aiogram.types.message.Message</pre>	<pre>as_reply_parameters()</pre>
method), 245	$(a iogram. types. in accessible \_message. In accessible Message$
<pre>answer_voice_pm() (aiogram.types.chat_join_request.Chat_join_</pre>	hatJoinReq <b>west</b> hod), 182
method), 101	as_reply_parameters()
AnswerCallbackQuery (class in	(aiogram.types.message.Message method),
aiogram.methods.answer_callback_query),	210
384	as_router() (aiogram.fsm.scene.Scene class method),
answered (aiogram.utils.callback_answer.CallbackAnswer	
property), 661	as_section() (in module aiogram.utils.formatting), 662
	AUD (aiogram.enums.currency.Currency attribute), 551
<pre>aiogram.methods.answer_inline_query), 521</pre>	AUDIO (aiogram.enums.content_type.ContentType at- tribute), 549
	AUDIO (aiogram.enums.inline_query_result_type.InlineQueryResultType
aiogram.methods.answer_pre_checkout_query),	attribute), 555
530	AUDIO (aiogram.enums.input_media_type.InputMediaType
AnswerShippingQuery (class in	attribute), 556
aiogram.methods.answer_shipping_query),	audio (aiogram.methods.send_audio.SendAudio at-
531 AnswerWebAppQuery (class in	tribute), 445
AnswerWebAppQuery (class in aiogram.methods.answer_web_app_query),	audio (aiogram.types.external_reply_info.ExternalReplyInfo attribute), 135
524	audio (aiogram.types.message.Message attribute), 206
ANY (aiogram.enums.content_type.ContentType at-	Audio (class in aiogram.types.audio), 20
tribute), 549	audio_duration(aiogram.types.inline_query_result_audio.InlineQuery.
api_url() (aiogram.client.telegram.TelegramAPIServer	attribute), 290
method), 14	audio_file_id(aiogram.types.inline_query_result_cached_audio.Inline
approve() (aiogram.types.chat_join_request.ChatJoinReq	
method), 63	audio_url(aiogram.types.inline_query_result_audio.InlineQueryResult_
ApproveChatJoinRequest (class in	attribute), 289
aiogram.methods.approve_chat_join_request),	auth_date (aiogram.utils.web_app.WebAppInitData at-
385	tribute), 656

author_signature (aiogram.types.message.Message attribute), 206	ge	base (aiogram.client.telegram.TelegramAPIServer attribute), 14
author_signature(aiogram.types.message_origin_chattribute), 259	hann	
	la cr. 4 A	
author_signature(aiogram.types.message_origin_chattribute), 260	ıaı.w	
**	at	aiogram.webhook.aiohttp_server), 629
		BaseSession (class in aiogram.client.session.base), 15
attribute), 37		BaseStorage (class in aiogram.fsm.storage.base), 602
available_reactions		BASKETBALL (aiogram.enums.dice_emoji.DiceEmoji at-
(aiogram.types.chat_full_info.ChatFullInfo		tribute), 554
attribute), 59		BASKETBALL (aiogram.types.dice.DiceEmoji attribute),
AZN (aiogram.enums.currency.Currency attribute), 551		133
В		BDT (aiogram.enums.currency.Currency attribute), 551
		BGN (aiogram.enums.currency.Currency attribute), 551
back() (aiogram.fsm.scene.SceneWizard method), 619		big_file_id (aiogram.types.chat_photo.ChatPhoto at-
background_custom_emoji_id		tribute), 131
(aiogram.types.chat.Chat attribute), 37		big_file_unique_id(aiogram.types.chat_photo.ChatPhoto
background_custom_emoji_id		attribute), 131
(aiogram.types.chat_full_info.ChatFullInfo		bio (aiogram.types.chat.Chat attribute), 37
attribute), 59		bio (aiogram.types.chat_full_info.ChatFullInfo at-
`	in	tribute), 60
<pre>aiogram.types.background_fill), 21</pre>		bio (aiogram.types.chat_join_request.ChatJoinRequest
BackgroundFillFreeformGradient (class i	in	attribute), 63
aiogram.types.background_fill_freeform_grad	lient	pirthdate (aiogram.types.chat.Chat attribute), 37
21		birthdate (aiogram.types.chat_full_info.ChatFullInfo
5	in	attribute), 59
<pre>aiogram.types.background_fill_gradient),</pre>		Birthdate (class in aiogram.types.birthdate), 25
22		BLOCKQUOTE (aiogram.enums.message_entity_type.MessageEntityType
BackgroundFillSolid (class i	in	attribute), 558
<pre>aiogram.types.background_fill_solid), 22</pre>		BlockQuote (class in aiogram.utils.formatting), 668
BackgroundType (class	in	BLUE (aiogram.enums.topic_icon_color.TopicIconColor
<pre>aiogram.types.background_type), 23</pre>		attribute), 561
5		BND (aiogram.enums.currency.Currency attribute), 551
<pre>aiogram.types.background_type_chat_theme),</pre>	7	BOB (aiogram.enums.currency.Currency attribute), 551
23		BOLD (aiogram.enums.message_entity_type.MessageEntityType
5 71	in	attribute), 557
<pre>aiogram.types.background_type_fill), 23</pre>		Bold (class in aiogram.utils.formatting), 666
BackgroundTypePattern (class i	in	boost (aiogram.types.chat_boost_updated.ChatBoostUpdated
<pre>aiogram.types.background_type_pattern),</pre>		attribute), 57
24		BOOST_ADDED (aiogram.enums.content_type.ContentType
BackgroundTypeWallpaper (class i	in	attribute), 550
<pre>aiogram.types.background_type_wallpaper), 25</pre>		boost_added (aiogram.types.message.Message at- tribute), 209
BAM (aiogram.enums.currency.Currency attribute), 551		boost_count(aiogram.types.chat_boost_added.ChatBoostAdded
ban() (aiogram.types.chat.Chat method), 49		attribute), 54
ban_sender_chat() (aiogram.types.chat.Chat method	<i>l</i> ),	boost_id (aiogram.types.chat_boost.ChatBoost at-
40	,	tribute), 54
BanChatMember (class	in	boost_id(aiogram.types.chat_boost_removed.ChatBoostRemoved
aiogram.methods.ban_chat_member), 386		attribute), 55
	in	boosts(aiogram.types.user_chat_boosts.UserChatBoosts
aiogram.methods.ban_chat_sender_chat),		attribute), 277
388		bot_administrator_rights
BANK_STATEMENT (aiogram.enums.encrypted_passport_attribute), 555	_elei	nent.Encr <b>\piearusspygaEkeyloga</b> rd_button_request_chat.KeyboardButtonRo attribute), 196

BOT_COMMAND (aiogram.enums.message_entity_type.Message_tribute), 557	
//	attribute), 562
bot_is_member (aiogram.types.keyboard_button_request_attribute), 196	_ <b>crus.жеуs»:<sub>(</sub>къпписи код</b> ие <b> (шову</b> тат.types.upaate.Opaate attribute), 361
	(aiogram.methods.edit_message_caption.EditMessageCaption
attribute), 201	
BotCommand (class in aiogram.types.bot_command), 26	attribute), 510
BotCommand (class in aiogram.utils.formatting), 666	business_connection_id
BotCommandScope (class in	(aiogram.methods.edit_message_live_location.EditMessageLiveL
aiogram.types.bot_command_scope), 26	attribute), 512
BotCommandScopeAllChatAdministrators (class in	
27	ministrator@jogram.methods.edit_message_media.EditMessageMedia attribute), 514
	business_connection_id
aiogram.types.bot_command_scope_all_group_c 27	hats), (aiogram.methods.edit_message_reply_markup.EditMessageReply_attribute), 515
BotCommandScopeAllPrivateChats (class in	business_connection_id
aiogram.types.bot_command_scope_all_private_ 28	
BotCommandScopeChat (class in	business_connection_id
<pre>aiogram.types.bot_command_scope_chat), 28</pre>	(aiogram.methods.get_business_connection.GetBusinessConnectiattribute), 415
BotCommandScopeChatAdministrators (class in	business_connection_id
aiogram.types.bot_command_scope_chat_admin.	istrators), (aiogram.methods.pin_chat_message.PinChatMessage attribute), 433
BotCommandScopeChatMember (class in	business_connection_id
aiogram.types.bot_command_scope_chat_membe	
BotCommandScopeDefault (class in	business_connection_id
aiogram.types.bot_command_scope_default),	(aiogram.methods.send_audio.SendAudio attribute), 445
	business_connection_id
aiogram.enums.bot_command_scope_type), 547	(aiogram.methods.send_chat_action.SendChatAction attribute), 448
	business_connection_id
aiogram.types.bot_description), 30	(aiogram.methods.send_contact.SendContact
BotName (class in aiogram.types.bot_name), 31	attribute), 449
	business_connection_id
aiogram.types.bot_short_description), 31	(aiogram.methods.send_dice.SendDice at-
bottom_color (aiogram.types.background_fill_gradient.B	
attribute), 22	business_connection_id
BOWLING (aiogram.enums.dice_emoji.DiceEmoji at- tribute), 554	(aiogram.methods.send_document.SendDocument attribute), 454
BOWLING (aiogram.types.dice.DiceEmoji attribute), 133	business_connection_id
BRL (aiogram.enums.currency.Currency attribute), 551	(aiogram.methods.send_game.SendGame
BufferedInputFile (class in aiogram.types.input_file),	attribute), 527
184, 566	business_connection_id
build() (aiogram.fsm.storage.base.DefaultKeyBuilder method), 602	(aiogram.methods.send_location.SendLocation attribute), 457
build() (aiogram.fsm.storage.base.KeyBuilder method),	business_connection_id
601 build() (aiogram.utils.media_group.MediaGroupBuilder	(aiogram.methods.send_media_group.SendMediaGroup attribute), 460
method), 672	business_connection_id
BUSINESS CONNECTION	(ajooram methods send message SendMessage

attribute), 462	business_opening_hours
business_connection_id (aiogram.methods.send_paid_media.SendPaidMe	(aiogram.types.chat_full_info.ChatFullInfo edia attribute), 59
attribute), 465	BusinessConnection (class in
business_connection_id	aiogram.types.business_connection), 31
(aiogram.methods.send_photo.SendPhoto attribute), 467	BusinessIntro (class in aiogram.types.business_intro), 32
business_connection_id	BusinessLocation (class in
(aiogram.methods.send_poll.SendPoll at-	aiogram.types.business_location), 32
tribute), 470	BusinessMessagesDeleted (class in
business_connection_id	aiogram.types.business_messages_deleted), 33
(aiogram.methods.send_sticker.SendSticker attribute), 373	BusinessOpeningHours (class in aiogram.types.business_opening_hours),
business_connection_id	33
(aiogram.methods.send_venue.SendVenue attribute), 473	BusinessOpeningHoursInterval (class in aiogram.types.business_opening_hours_interval),
business_connection_id	34
(aiogram.methods.send_video.SendVideo attribute), 476	button (aiogram.methods.answer_inline_query.AnswerInlineQuery attribute), 523
business_connection_id	$\verb button_text  (a iogram. types. web\_app\_data. WebAppData $
(aiogram.methods.send_video_note.SendVideoNo	
attribute), 479	buttons (aiogram.utils.keyboard.InlineKeyboardBuilder
business_connection_id	property), 645
(aiogram.methods.send_voice.SendVoice	buttons (aiogram.utils.keyboard.ReplyKeyboardBuilder
<pre>attribute), 482 business_connection_id</pre>	property), 646
(aiogram.methods.stop_message_live_location.St	BYN (aiogram.enums.currency.Currency attribute), 551
attribute), 519	C
business_connection_id	cache_time(aiogram.methods.answer_callback_query.AnswerCallbackQ
$(a iogram.methods.stop\_poll.StopPoll\ attribute),$	attribute), 384
520	${\tt cache\_time} \ (a iogram.methods.answer\_inline\_query. AnswerInlineQuery$
business_connection_id	attribute), 522
attribute), 506	At Mense (aiogram.utils.callback_answer.CallbackAnswer property), 661
business_connection_id	CAD (aiogram.enums.currency.Currency attribute), 551
(aiogram.types.business_messages_deleted.Busin attribute), 33	essMrssesesesesesesesesesesesesesesesese
business_connection_id	callback_data (aiogram.types.inline_keyboard_button.InlineKeyboardBi
(aiogram.types.message.Message attribute),	attribute), 183
205	callback_game (aiogram.types.inline_keyboard_button.InlineKeyboardButton.
business_intro (aiogram.types.chat.Chat attribute),	attribute), 184
37	CALLBACK_QUERY (aiogram.enums.update_type.UpdateType
business_intro(aiogram.types.chat_full_info.ChatFullIn	
attribute), 59	callback_query (aiogram.types.update.Update at-
business_location (aiogram.types.chat.Chat at- tribute), 37	tribute), 361
business_location(aiogram.types.chat_full_info.ChatF	• • • • • • • • • • • • • • • • • • • •
attribute), 59	callback_query_without_state
BUSINESS_MESSAGE (aiogram.enums.update_type.UpdateTattribute), 562	619
business_message (aiogram.types.update.Update at- tribute), 361	CallbackAnswer (class in aiogram.utils.callback_answer), 660
business_opening_hours (aiogram.types.chat.Chat attribute), 37	CallbackAnswerException, 626

attribute), 105

attribute), 436

```
CallbackAnswerMiddleware
                                                                                                                                                            (class
                                                                                                                                                                                                                     in
                                                                                                                                                                                                                                                                             attribute), 53
                                      aiogram.utils.callback_answer), 660
                                                                                                                                                                                                                                      can_edit_stories(aiogram.types.chat_member_administrator.ChatMem
                                                                                                                                                                                                                                                                             attribute), 105
CallbackData (class in aiogram.filters.callback_data),
                                                                                                                                                                                                                                      can_invite_users(aiogram.methods.promote_chat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_
CallbackGame (class in aiogram.types.callback_game),
                                                                                                                                                                                                                                                                             attribute), 436
                                                                                                                                                                                                                                       can_invite_users(aiogram.types.chat_administrator_rights.ChatAdmin
CallbackQuery
                                                                                                                                                                                                                                                                             attribute), 52
                                                                                                                                (class
                                                                                                                                                                                                                     in
                                      aiogram.types.callback_query), 34
                                                                                                                                                                                                                                       can_invite_users(aiogram.types.chat_member_administrator.ChatMem
CallbackQueryHandler
                                                                                                                                                 (class
                                                                                                                                                                                                                                                                             attribute), 105
                                                                                                                                                                                                                     in
                                      aiogram.handlers.callback_query), 639
                                                                                                                                                                                                                                       can_invite_users(aiogram.types.chat_member_restricted.ChatMember_
can_add_web_page_previews
                                                                                                                                                                                                                                                                             attribute), 109
                                      (aiogram.types.chat_member_restricted.ChatMembanRaxtwätael_users (aiogram.types.chat_permissions.ChatPermissions
                                     attribute), 109
                                                                                                                                                                                                                                                                             attribute), 131
can_add_web_page_previews
                                                                                                                                                                                                                                       can_join_groups (aiogram.types.user.User attribute),
                                     (aiogram.types.chat_permissions.ChatPermissions
                                     attribute), 131
                                                                                                                                                                                                                                       can\_manage\_chat (aiogram.methods.promote\_chat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_m
can_be_edited(aiogram.types.chat_member_administrator.ChatMentbeib4dm)ujAisTrator
                                      attribute), 104
                                                                                                                                                                                                                                      can_manage_chat (aiogram.types.chat_administrator_rights.ChatAdminis
can_change_info(aiogram.methods.promote_chat_member.Promote@thiltMe)mber
                                      attribute), 435
                                                                                                                                                                                                                                       can_manage_chat (aiogram.types.chat_member_administrator.ChatMemb
can\_change\_info (\it aiogram.types.chat\_administrator\_rights.ChatAd \textit{ntinilstrato}) on \textit{Rights} is the control of the contro
                                      attribute), 52
                                                                                                                                                                                                                                      can_manage_topics (aiogram.methods.promote_chat_member.PromoteClass)
can_change_info(aiogram.types.chat_member_administrator.ChatMtmibuteAdAishistrator
                                      attribute), 105
                                                                                                                                                                                                                                       can_manage_topics (aiogram.types.chat_administrator_rights.ChatAdmi
can_change_info(aiogram.types.chat_member_restricted.ChatMembteribester)cfed
                                     attribute), 109
                                                                                                                                                                                                                                       can_manage_topics (aiogram.types.chat_member_administrator.ChatMe.
can_change_info(aiogram.types.chat_permissions.ChatPermissionsattribute), 105
                                                                                                                                                                                                                                       can_manage_topics (aiogram.types.chat_member_restricted.ChatMembe
                                     attribute), 131
can_connect_to_business (aiogram.types.user.User
                                                                                                                                                                                                                                                                             attribute), 109
                                     attribute), 276
                                                                                                                                                                                                                                       can_manage_topics (aiogram.types.chat_permissions.ChatPermissions
can_delete_messages
                                                                                                                                                                                                                                                                             attribute), 131
                                      (aiogram.methods.promote\_chat\_member.PromoteChatMember
                                     attribute), 435
                                                                                                                                                                                                                                                                             attribute), 435
can_delete_messages
                                       (aiogram.types.chat_administrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatA
                                                                                                                                                                                                                                                                             (aiogram.types.chat_administrator_rights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministratorRights.ChatAdministr
                                     attribute), 52
can_delete_messages
                                                                                                                                                                                                                                                                              attribute), 52
                                      (aiogram.types.chat_member_administrator.ChatMandmaAdgenisivdeor_chats
                                      attribute), 104
                                                                                                                                                                                                                                                                              (aiogram.types.chat_member_administrator.ChatMemberAdminis
can_delete_stories (aiogram.methods.promote_chat_member.Promote@Ghe);Member
                                                                                                                                                                                                                                      can_pin_messages (aiogram.methods.promote_chat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_memb
                                      attribute), 436
can_delete_stories(aiogram.types.chat_administrator_rights.ChattAxtbrita)tratorRights
                                                                                                                                                                                                                                      can_pin_messages (aiogram.types.chat_administrator_rights.ChatAdmin
                                      attribute), 53
can_delete_stories(aiogram.types.chat_member_administrator.ChattalbanabetaAdministrator
                                                                                                                                                                                                                                       can_pin_messages (aiogram.types.chat_member_administrator.ChatMem
                                     attribute), 105
can_edit_messages (aiogram.methods.promote_chat_member.Promotte@hateMember
                                     attribute), 436
                                                                                                                                                                                                                                      can_pin_messages (aiogram.types.chat_member_restricted.ChatMember_
can_edit_messages (aiogram.types.chat_administrator_rights.ChatAtdrilibinitxi);at@PRights
                                     attribute), 53
                                                                                                                                                                                                                                      can_pin_messages (aiogram.types.chat_permissions.ChatPermissions
can_edit_messages (aiogram.types.chat_member_administrator.ChattYfbmulee)rAdininistrator
```

734 Index

can\_edit\_stories(aiogram.methods.promote\_chat\_member.Promo**teEthatte1**)cmbbbr

can\_edit\_stories (aiogram.types.chat\_administrator\_rights.ChatAalmiibistedut&Rights

 $\verb|can_post_messages|| (a iogram.methods.promote\_chat\_member.PromoteC$ 

can\_post\_messages (aiogram.types.chat\_administrator\_rights.ChatAdmi

caption(aiogram.types.inline\_query\_result\_cached\_gif.InlineQueryResult\_

```
can_post_messages (aiogram.types.chat_member_administrator.ChatMboulee);Administrator
              attribute), 105
                                                                                     can_send_photos(aiogram.types.chat_permissions.ChatPermissions
can_post_stories (aiogram.methods.promote chat member.Promote Chatter and Der
              attribute), 436
                                                                                     can_send_polls (aiogram.types.chat_member_restricted.ChatMemberRes
can_post_stories (aiogram.types.chat_administrator_rights.ChatAalmiibisteoutoftPaights
              attribute), 53
                                                                                     can_send_polls(aiogram.types.chat permissions.ChatPermissions
can_post_stories (aiogram.types.chat member administrator.ChatMribiber)Administrator
                                                                                      can_send_video_notes
              attribute), 105
can_promote_members
                                                                                                    (aiogram.types.chat_member_restricted.ChatMemberRestricted
              (aiogram.methods.promote_chat_member.PromoteChatMemtteribute), 108
              attribute), 435
                                                                                      can_send_video_notes
can_promote_members
                                                                                                    (aiogram.types.chat_permissions.ChatPermissions
              (aiogram.types.chat_administrator_rights.ChatAdministratoaRightse), 130
                                                                                      can_send_videos(aiogram.types.chat_member_restricted.ChatMemberR
              attribute), 52
can_promote_members
                                                                                                    attribute), 108
              (aiogram.types.chat_member_administrator.ChatMambseAdminideoschatogram.types.chat_permissions.ChatPermissions
              attribute), 105
                                                                                                    attribute), 130
can_read_all_group_messages
                                                                                      can_send_voice_notes
              (aiogram.types.user.User attribute), 276
                                                                                                    (aiogram.types.chat_member_restricted.ChatMemberRestricted
can_reply (aiogram.types.business_connection.BusinessConnection attribute), 109
              attribute), 32
                                                                                     can_send_voice_notes
can_restrict_members
                                                                                                    (aiogram.types.chat_permissions.ChatPermissions
              (aiogram.methods.promote_chat_member.PromoteChatMemdteribute), 130
              attribute), 435
                                                                                      can_set_sticker_set
                                                                                                                                    (aiogram.types.chat.Chat
can_restrict_members
                                                                                                    attribute), 37
              (aiogram.types.chat_administrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatA
              attribute), 52
                                                                                                    (aiogram.types.chat_full_info.ChatFullInfo
can_restrict_members
                                                                                                    attribute), 61
              (aiogram.types.chat_member_administrator.ChatMaptbieoAdministratorethods.copy_message.CopyMessage
              attribute), 105
                                                                                                    attribute), 393
can_send_after(aiogram.utils.web_app.WebAppInitDatacaption(aiogram.methods.edit_message_caption.EditMessageCaption
              attribute), 656
                                                                                                    attribute), 510
can_send_audios (aiogram.types.chat_member_restricted Captt Many bior Rasmintethods.send_animation.SendAnimation
              attribute), 108
                                                                                                    attribute), 443
can_send_audios (aiogram.types.chat_permissions.ChatPeaptsions (aiogram.methods.send_audio.SendAudio at-
              attribute), 130
                                                                                                    tribute), 445
can_send_documents(aiogram.types.chat member restrictad:Glan MièndrenResethadd.send document.SendDocument
              attribute), 108
                                                                                                    attribute), 454
can_send_documents (aiogram.types.chat_permissions.Clauftingram.methods.send_paid_media.SendPaidMedia
              attribute), 130
                                                                                                    attribute), 465
can_send_messages (aiogram.types.chat_member_restrictedqtfliatMe(abegRest.riathbds.send_photo.SendPhoto at-
              attribute), 108
                                                                                                    tribute), 467
can_send_messages (aiogram.types.chat_permissions.ChatRotribations
                                                                                                            (aiogram.methods.send_video.SendVideo
              attribute), 130
                                                                                                    attribute), 477
can_send_other_messages
                                                                                                             (aiogram.methods.send_voice.SendVoice
                                                                                      caption
              (aiogram.types.chat_member_restricted.ChatMemberRestriattalibute), 482
              attribute), 109
                                                                                     caption (aiogram.types.inline_query_result_audio.InlineQueryResultAudi
can_send_other_messages
                                                                                                    attribute), 289
              (aiogram.types.chat_permissions.ChatPermissions.caption (aiogram.types.inline_query_result_cached_audio.InlineQueryRe
              attribute), 130
                                                                                                    attribute), 292
can_send_paid_media
                                                                                      caption(aiogram.types.inline_query_result_cached_document.InlineQuer
                                                                                                    attribute), 294
              (aiogram.types.chat_full_info.ChatFullInfo
```

Index 735

can\_send\_photos (aiogram.types.chat\_member\_restricted.ChatMemberiberib);cledb

attribute), 60

```
caption (aiogram.types.inline_query_result_cached_mpeg4_gif.Inlina@ubmtResultCachedMpeg4Gif
                                                                                   caption_entities(aiogram.types.inline_query_result_cached_gif.Inline
             attribute), 298
caption (aiogram.types.inline_query_result_cached_photo.InlineQuartyRbsult)CaOkedPhoto
                                                                                   caption_entities(aiogram.types.inline_query_result_cached_mpeg4_g
             attribute), 300
caption (aiogram.types.inline_query_result_cached_video.InlineQueartRibuth)CaeNeedVideo
             attribute), 304
                                                                                   caption_entities(aiogram.types.inline query result cached photo.Inli
caption (aiogram.types.inline_query_result_cached_voice.InlineQuentiRidmuleCachedVoice
                                                                                   caption_entities(aiogram.types.inline_query_result_cached_video.Inli
              attribute), 306
caption (aiogram.types.inline_query_result_document.InlineQueryRastrilDoc)m0At
                                                                                   caption_entities(aiogram.types.inline_query_result_cached_voice.Inli
             attribute), 310
caption(aiogram.types.inline_query_result_gif.InlineQueryResultGifittribute), 306
                                                                                   caption_entities(aiogram.types.inline_query_result_document.InlineQ
             attribute), 313
caption (aiogram.types.inline_query_result_mpeg4_gif.InlineQueryRutxiltIMp);g44Gif
             attribute), 317
                                                                                   caption_entities(aiogram.types.inline_query_result_gif.InlineQueryRe
caption(aiogram.types.inline_query_result_photo.InlineQueryResultPhribate), 313
                                                                                    caption_entities(aiogram.types.inline_query_result_mpeg4_gif.Inlineg
              attribute), 319
caption (aiogram.types.inline_query_result_video.InlineQueryResultMtdibute), 317
             attribute), 323
                                                                                   caption_entities(aiogram.types.inline_query_result_photo.InlineQuer
caption (aiogram.types.inline_query_result_voice.InlineQueryResultMoideute), 319
                                                                                   {\tt caption\_entities} \ (a iogram. types. in line\_query\_result\_video. In line Query\_result\_video. In line Query\_r
             attribute), 324
caption(aiogram.types.input_media_animation.InputMediaAnimatiattribute), 323
             attribute), 186
                                                                                   caption_entities (aiogram.types.inline_query_result_voice.InlineQuery
{\tt caption}\ (aiogram.types.input\_media\_audio.InputMediaAudio
                                                                                                 attribute), 325
              attribute), 187
                                                                                   caption_entities(aiogram.types.input media animation.InputMediaAr
caption (aiogram.types.input_media_document.InputMediaDocumentttribute), 186
             attribute), 189
                                                                                   caption_entities(aiogram.types.input_media_audio.InputMediaAudio
{\tt caption} \ (aiogram.types.input\_media\_photo.InputMediaPhoto
                                                                                                 attribute), 188
                                                                                   caption_entities(aiogram.types.input_media_document.InputMediaDa
             attribute), 189
caption(aiogram.types.input_media_video.InputMediaVideo
                                                                                                 attribute), 189
             attribute), 191
                                                                                   caption_entities(aiogram.types.input_media_photo.InputMediaPhoto
caption (aiogram.types.message.Message attribute), 207
                                                                                                 attribute), 190
caption_entities(aiogram.methods.copy_message.Copyddpstrign_entities(aiogram.types.input_media_video.InputMediaVideo
             attribute), 393
                                                                                                 attribute), 191
caption_entities (aiogram.methods.edit_message_caption_tithde.sangicCinption (aiogram.types.message.Message
             attribute), 510
                                                                                                 attribute), 207
caption_entities(aiogram.methods.send_animation.SenQlaSHIFIAGi(uniogram.enums.message_entity_type.MessageEntityType
             attribute), 443
                                                                                                 attribute), 557
caption_entities(aiogram.methods.send_audio.SendAudiashTag(class in aiogram.utils.formatting), 666
             attribute), 445
                                                                                   certificate(aiogram.methods.set_webhook.SetWebhook
caption_entities(aiogram.methods.send_document.SendDocumentttribute), 544
             attribute), 455
                                                                                   CHANNEL (aiogram.enums.chat type.ChatType attribute),
{\tt caption\_entities} \ (a iogram.methods.send\_paid\_media.SendPaidM\it \it E\!A\!B\!a
                                                                                   CHANNEL (aiogram.enums.message_origin_type.MessageOriginType
             attribute), 465
{\tt caption\_entities} \ (a iogram.methods.send\_photo.SendPhoto
                                                                                                 attribute), 558
                                                                                   CHANNEL_CHAT_CREATED
             attribute), 467
caption_entities(aiogram.methods.send_video.SendVideo
                                                                                                 (aiogram.enums.content_type.ContentType
             attribute), 477
                                                                                                 attribute), 550
caption_entities(aiogram.methods.send_voice.SendVoichannel_chat_created
             attribute), 482
                                                                                                 (aiogram.types.message.Message
                                                                                                                                                      attribute),
caption_entities(aiogram.types.inline_query_result_audio.Inline@neryResultAudio
             attribute), 290
                                                                                   CHANNEL_POST (aiogram.enums.update_type.UpdateType
caption_entities(aiogram.types.inline query result cached audiathibimaeQuientResultCachedAudio
             attribute), 292
                                                                                   channel_post (aiogram.types.update.Update attribute),
caption_entities(aiogram.types.inline query result cached document.InlineQueryResultCachedDocument
```

```
CHAT (aiogram.enums.bot_command_scope_type.BotCommandScopeTatpeibute), 390
                  attribute), 547
                                                                                                                 {\tt chat\_id}\ (aiogram.methods.close\_general\_forum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_to
CHAT (aiogram.enums.message origin type.MessageOriginType
                                                                                                                                    attribute), 391
                                                                                                                 \verb|chat_id| (a iogram.methods.copy_message.CopyMessage|
                  attribute), 558
CHAT (aiogram.fsm.strategy.FSMStrategy attribute), 603
                                                                                                                                    attribute), 392
chat (aiogram.types.business messages deleted.BusinessMehangestDelatedam.methods.copy messages.CopyMessages
                  attribute), 33
                                                                                                                                    attribute), 394
attribute), 55
                                                                                                                                    attribute), 396
chat (aiogram.types.chat_boost_updated.ChatBoostUpdatedthat_id (aiogram.methods.create_chat_subscription_invite_link.CreateCh
                  attribute), 57
                                                                                                                                    attribute), 398
chat (aiogram.types.chat_join_request.ChatJoinRequest chat_id (aiogram.methods.create_forum_topic.CreateForumTopic
                  attribute), 62
                                                                                                                                    attribute), 399
chat (aiogram.types.chat_member_updated.ChatMemberUpdhatedid (aiogram.methods.decline_chat_join_request.DeclineChatJoinRed
                  attribute), 109
                                                                                                                                    attribute), 400
chat (aiogram.types.external_reply_info.ExternalReplyInfo chat_id (aiogram.methods.delete_chat_photo.DeleteChatPhoto
                  attribute), 135
                                                                                                                                    attribute), 402
chat (aiogram.types.giveaway_winners.GiveawayWinners chat_id (aiogram.methods.delete_chat_sticker_set.DeleteChatStickerSet
                  attribute), 142
                                                                                                                                    attribute), 403
chat (aiogram.types.inaccessible_message.InaccessibleMesshet_id (aiogram.methods.delete_forum_topic.DeleteForumTopic
                  attribute), 143
                                                                                                                                    attribute), 404
chat (aiogram.types.message.Message attribute), 205
                                                                                                                 chat_id(aiogram.methods.delete_message.DeleteMessage
chat (aiogram.types.message_origin_channel.MessageOriginChannehttribute), 507
                                                                                                                 chat_id(aiogram.methods.delete messages.DeleteMessages
                   attribute), 259
chat (aiogram.types.message_reaction_count_updated.MessageReactionflute) Libitated
                                                                                                                 \verb|chat_id| (a iogram.methods.edit\_chat_invite\_link.EditChatInviteLink|)|
                  attribute), 261
chat (aiogram.types.message_reaction_updated.MessageReactionUpdateibute), 406
                                                                                                                 chat_id (aiogram.methods.edit_chat_subscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.EditChatSubscription_invite_link.E
                  attribute), 262
chat (aiogram.types.story.Story attribute), 273
                                                                                                                                    attribute), 408
chat (aiogram.utils.web_app.WebAppInitData attribute),
                                                                                                               chat_id(aiogram.methods.edit_forum_topic.EditForumTopic
                   656
                                                                                                                                    attribute), 409
Chat (class in aiogram.types.chat), 36
                                                                                                                 chat_id(aiogram.methods.edit_general_forum_topic.EditGeneralForumTe
CHAT_ADMINISTRATORS
                                                                                                                                    attribute), 410
                   (aiogram.enums.bot_command_scope_type.BotCochnatindls(apiefyma.methods.edit_message_caption.EditMessageCaption
                  attribute), 547
                                                                                                                                    attribute), 510
CHAT_BACKGROUND_SET
                                                                                                                 chat_id(aiogram.methods.edit_message_live_location.EditMessageLiveLo
                  (aiogram.enums.content type.ContentType
                                                                                                                                    attribute), 512
                  attribute), 550
                                                                                                                 chat_id(aiogram.methods.edit_message_media.EditMessageMedia
chat_background_set
                                                                                                                                    attribute), 514
                  (aiogram.types.message.Message
                                                                                                                 chat_id(aiogram.methods.edit_message_reply_markup.EditMessageReply
                                                                                         attribute),
                                                                                                                                    attribute), 515
CHAT_BOOST (aiogram.enums.update_type.UpdateType
                                                                                                                 chat_id (aiogram.methods.edit_message_text.EditMessageText
                  attribute), 562
                                                                                                                                    attribute), 517
                                                                                                                 \verb|chat_id| (a iogram.methods.export\_chat\_invite\_link. ExportChatInviteLink|) \\
chat_boost (aiogram.types.update.Update attribute),
                                                                                                                                    attribute), 411
chat_has_username(aiogram.types.keyboard_button_requalisa_tchicd.(Kaioyboard Button Rejonesa Chatessage.ForwardMessage
                  attribute), 196
                                                                                                                                    attribute), 412
chat_id (aiogram.methods.approve_chat_join_request.AppednætCliad.JainReamestethods.forward_messages.ForwardMessages
                                                                                                                                    attribute), 414
                   attribute), 385
chat_id (aiogram.methods.ban_chat_member.BanChatMembert_id (aiogram.methods.get_chat.GetChat attribute),
                   attribute), 386
chat_id(aiogram.methods.ban chat sender chat.BanChatSwatleid(atogram.methods.get chat administrators.GetChatAdministrator
                   attribute), 388
                                                                                                                                    attribute), 417
```

chat\_id(aiogram.methods.close forum topic.CloseForumEbpait\_id(aiogram.methods.get chat member.GetChatMember

tribute), 470

attribute), 418 tribut	e), 373
chat_id(aiogram.methods.get_chat_member_count.GetChahlember(Cain) attribute), 419 tribut	tram.methods.send_venue.SendVenue at- e), 473
chat_id(aiogram.methods.get_chat_menu_button.GetChat@hatuBudton attribute), 420 attrib	(aiogram.methods.send_video.SendVideo ute), 476
chat_id(aiogram.methods.get_game_high_scores.GetGameHigh_Scotteriognattribute), 526 attrib	am.methods.send_video_note.SendVideoNote ute), 479
chat_id(aiogram.methods.get_user_chat_boosts.GetUser@hatBoists	(aiogram.methods.send_voice.SendVoice ute), 482
chat_id(aiogram.methods.hide_general_forum_topic.HideCharte_ritF(ariogi	
chat_id (aiogram.methods.leave_chat.LeaveChat chat_id(aiogram.methods.leave_chat.LeaveChat chat_id(aiogram.methods.leave_chat.LeaveChat.	
chat_id(aiogram.methods.pin_chat_message.PinChatMessaget_id(aiogram.methods.pin_chat_message.PinChatMessaget_id(aiogram.methods.pin_chat_message.PinChatMessaget_id(aiogram.methods.pin_chat_messaget.PinChatMessaget_id(aiogram.methods.pin_chat_messaget.PinChatMessaget.Pin	
chat_id(aiogram.methods.promote_chat_member.PromoteChart_Medr(briogram.methods.promote_chat_member.PromoteChart_Medr(briogram.methods.promote_chat_member.PromoteChart_Medr(briogram.methods.promote_chat_member.PromoteChart_Medr(briogram.methods.promote_chat_member.PromoteChart_Medr(briogram.methods.promote_chat_member.PromoteChart_Medr(briogram.methods.promote_chat_member.PromoteChart_Medr(briogram.methods.promote_chat_member.PromoteChart_Medr(briogram.methods.promote_chat_member.Promote_chat_member.PromoteChart_Medr(briogram.methods.promote_chat_member.Promote	
chat_id(aiogram.methods.reopen_forum_topic.ReopenForumaffopid(aiogram.methods.reopen_forum_topic.ReopenForumaffopid(aiogram.methods.reopen_forum_topic.ReopenForumaffopid(aiogram.methods.reopen_forum_topic.ReopenForumaffopid(aiogram.methods.reopen_forum_topic.ReopenForumaffopid(aiogram.methods.reopen_forum_topic.ReopenForumaffopid(aiogram.methods.reopen_forum_topic.ReopenForumaffopid(aiogram.methods.reopen_forum_topic.ReopenForumaffopid(aiogram.methods.reopen_forum_topic.ReopenForumaffopid(aiogram.methods.reopen_forum_topic.ReopenForumaffopid(aiogram.methods.reopen_forum_topic.ReopenForumaffopid(aiogram.methods.reopen_forum_topic.ReopenForumaffopid(aiogram.methods.reopen_forum_topic.Reopen_forum_topic.Reopen_forumaffopid(aiogram.methods.reopen_forum_topic.Reopen_forum_topic.Reopen_forum_topic.Reopen_forum_topic.Reopen_forumaffopid(aiogram.methods.reopen_forum_topic.Reopen_for	
chat_id(aiogram.methods.reopen_general_forum_topic.Redpart_Gedtenidfe	
chat_id(aiogram.methods.restrict_chat_member.RestrictChhettleindber (aio attribute), 439 attrib	ogram.methods.set_chat_title.SetChatTitle ute), 491
chat_id(aiogram.methods.revoke_chat_invite_link.Revoke <b>Chat</b> Invi <b>t</b> e\text{Link} attribute), 440 attrib	am.methods.set_game_score.SetGameScore ute), 529
<pre>chat_id(aiogram.methods.send_animation.SendAnimationchat_id(aiogramtibute), 442</pre> <pre>attribute</pre>	ram.methods.set_message_reaction.SetMessageReaction ute), 492
<pre>chat_id (aiogram.methods.send_audio.SendAudio at- chat_id(aiogram.tribute), 445</pre>	am.methods.stop_message_live_location.StopMessageLiveL ute), 519
<pre>chat_id (aiogram.methods.send_chat_action.SendChatActionat_id (aiogram.tribute), 448</pre> 520	ram.methods.stop_poll.StopPoll attribute),
	am.methods.unban_chat_member.UnbanChatMember ute), 499
	am.methods.unban_chat_sender_chat.UnbanChatSenderCh ute), 500
<pre>chat_id(aiogram.methods.send_document.SendDocument chat_id(aiogramtibute), 454</pre> <pre>attribute</pre>	am.methods.unhide_general_forum_topic.UnhideGeneralFo ute), 501
chat_id (aiogram.methods.send_game.SendGame chat_id(aiogramtribute), 527 attrib	am.methods.unpin_all_chat_messages.UnpinAllChatMessagute), 502
	am.methods.unpin_all_forum_topic_messages.UnpinAllForuute), 503
	am.methods.unpin_all_general_forum_topic_messages.Unp ute), 504
chat_id(aiogram.methods.send_media_group.SendMedia@hatp_id(aiogramtribute), 459 attrib	am.methods.unpin_chat_message.UnpinChatMessage ute), 506
	am.types.bot_command_scope_chat.BotCommandScopeChaute), 28
chat_id(aiogram.methods.send_paid_media.SendPaidMed <b>th</b> at_id(aiogramtibute), 464 attrib	am.types.bot_command_scope_chat_administrators.BotConute), 29
	am.types.bot_command_scope_chat_member.BotCommand ute), 29
<pre>chat_id (aiogram.methods.send_poll.SendPoll at- chat_id (aiogram.methods.send_poll.Send.send_poll.Send.send.send.send.send.send.send.send.s</pre>	gram.types.chat_shared.ChatShared at-

738 Index

tribute), 132

attribute), 271	ChatBoostSourcePremium (class	in
<pre>chat_instance(aiogram.types.callback_query.CallbackQ</pre>	Query aiogram.types.chat_boost_source_premium), 57	
$\verb chat_instance   (a iogram.utils.web_app.WebAppInitData  $		in
attribute), 656	<pre>aiogram.enums.chat_boost_source_type),</pre>	
chat_is_channel (aiogram.types.keyboard_button_reque		
attribute), 196	•	in
chat_is_created(aiogram.types.keyboard_button_reque		`
attribute), 196	ChatFullInfo (class in aiogram.types.chat_full_info	9),
chat_is_forum(aiogram.types.keyboard_button_request_ attribute), 196		in
CHAT_JOIN_REQUEST (aiogram.enums.update_type.Update	`	ırı
attribute), 562		in
chat_join_request (aiogram.types.update.Update at-	aiogram.types.chat_join_request), 62	ııı
tribute), 362	ChatLocation (class in aiogram.types.chat_location	ı).
CHAT_MEMBER (aiogram.enums.bot_command_scope_type attribute), 547	BotCommahdScopeType	
CHAT_MEMBER (aiogram.enums.update_type.UpdateType	ChatMember (class in aiogram.types.chat_member), 10 ChatMemberAdministrator (class	in
attribute), 562	aiogram.types.chat_member_administrator),	ırı
chat_member (aiogram.types.update.Update attribute),	103	
362		in
CHAT_SHARED (aiogram.enums.content_type.ContentType	aiogram.types.chat_member_banned), 106	
attribute), 550		in
chat_shared (aiogram.types.message.Message at-	aiogram.types.chat_member_left), 106	
tribute), 208		in
CHAT_TOPIC (aiogram.fsm.strategy.FSMStrategy at-	aiogram.types.chat_member_member), 107	
tribute), 603	ChatMemberOwner (class	in
chat_type (aiogram.types.inline_query.InlineQuery at-	aiogram.types.chat_member_owner), 107	_
tribute), 285	`	in
chat_type (aiogram.utils.web_app.WebAppInitData at- tribute), 656	aiogram.types.chat_member_restricted), 108	
ChatAction (class in aiogram.enums.chat_action), 547		in
ChatActionMiddleware (class in	aiogram.enums.chat_member_status), 548	_
aiogram.utils.chat_action), 653	• ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` `	in
ChatActionSender (class in aiogram.utils.chat_action),	aiogram.types.chat_member_updated), 109	
652	•	in
ChatAdministratorRights (class in aiogram.types.chat_administrator_rights),	aiogram.filters.chat_member_updated), 581 ChatPermissions (class	in
51	aiogram.types.chat_permissions), 130	ııı
ChatBackground (class in	ChatPhoto (class in aiogram.types.chat_photo), 131	
aiogram.types.chat_background), 53	chats (aiogram.types.giveaway.Giveaway attribute), 14	40
ChatBoost (class in aiogram.types.chat_boost), 54	ChatShared (class in aiogram.types.chat_shared), 132	
ChatBoostAdded (class in	ChatType (class in aiogram.enums.chat_type), 549	
aiogram.types.chat_boost_added), 54	check_flags() (in module aiogram.dispatcher.flags	5),
ChatBoostRemoved (class in	628	
aiogram.types.chat_boost_removed), 55	check_response() (aiogram.client.session.base.Base.	Sessior
ChatBoostSource (class in	method), 15	
<pre>aiogram.types.chat_boost_source), 55</pre>	check_webapp_signature() (in modu	le
ChatBoostSourceGiftCode (class in	aiogram.utils.web_app), 654	
aiogram.types.chat_boost_source_gift_code),	CHF (aiogram.enums.currency.Currency attribute), 551	
56	CHIN (aiogram.enums.mask_position_point.MaskPosition_	onPoin
ChatBoostSourceGiveaway (class in	attribute), 557	. •
aiogram.types.chat_boost_source_giveaway),	CHOOSE_STICKER (aiogram.enums.chat_action.ChatAct	tion
56	attribute) 548	

${\tt choose\_sticker()} \ (a iogram.utils.chat\_action.ChatAction) \\$	
class method), 652	attribute), 208
CHOSEN_INLINE_RESULT	ConstI18nMiddleware (class in
(aiogram.enums.update_type.UpdateType	aiogram.utils.i18n.middleware), 649
attribute), 562	CONTACT (aiogram.enums.content_type.ContentType at-
chosen_inline_result (aiogram.types.update.Update	tribute), 549
attribute), 361	CONTACT (aiogram.enums.inline_query_result_type.InlineQueryResultType
ChosenInlineResult (class in	attribute), 555
aiogram.types.chosen_inline_result), 284 city (aiogram.types.shipping_address.ShippingAddress	contact (aiogram.types.external_reply_info.ExternalReplyInfo attribute), 135
attribute), 343	contact (aiogram.types.message.Message attribute), 207
clear_data() (aiogram.fsm.scene.SceneWizard	Contact (class in aiogram.types.contact), 132
method), 619	content_type (aiogram.types.message.Message prop-
ClientDecodeError, 627	erty), 210
Close (class in aiogram.methods.close), 389	ContentType (class in aiogram.enums.content_type),
close() (aiogram.client.session.base.BaseSession	549
method), 15	COP (aiogram.enums.currency.Currency attribute), 552
close() (aiogram.fsm.scene.ScenesManager method), 618	copy() (aiogram.utils.keyboard.InlineKeyboardBuilder method), 645
close() (aiogram.fsm.storage.base.BaseStorage method), 603	copy() (aiogram.utils.keyboard.ReplyKeyboardBuilder method), 646
close() (aiogram.webhook.aiohttp_server.SimpleRequestl	
method), 630	247
close_date (aiogram.methods.send_poll.SendPoll attribute), 471	CopyMessage (class in aiogram.methods.copy_message), 392
close_date (aiogram.types.poll.Poll attribute), 266	CopyMessages (class in
CloseForumTopic (class in	aiogram.methods.copy_messages), 394
aiogram.methods.close_forum_topic), 390	correct_option_id(aiogram.methods.send_poll.SendPoll
CloseGeneralForumTopic (class in	attribute), 471
aiogram.methods.close_general_forum_topic), 391	correct_option_id (aiogram.types.poll.Poll attribute), 266
attribute), 34	<u>coventedyBosides(xOppercingHyperssInipepinel_</u> address.ShippingAddress attribute), 343
CLP (aiogram.enums.currency.Currency attribute), 551	country_codes (aiogram.types.giveaway.Giveaway at-
CNY (aiogram.enums.currency.Currency attribute), 552	tribute), 140
CODE (aiogram.enums.message_entity_type.MessageEntity]	
attribute), 558	create_invite_link() (aiogram.types.chat.Chat
Code (class in aiogram.utils.formatting), 667	method), 43
color (aiogram.types.background_fill_solid.BackgroundFi	
attribute), 22	aiogram.utils.deep_linking), 673
colors (aiogram.types.background_fill_freeform_gradient.	0 1 = 07·
attribute), 22	aiogram.methods.create_chat_invite_link),
command (aiogram.filters.command.CommandObject at-	396
tribute), 580	CreateChatSubscriptionInviteLink (class in
command (aiogram.types.bot_command.BotCommand at-	aiogram.methods.create_chat_subscription_invite_link),
tribute), 26	398
Command (class in aiogram.filters.command), 580	CreateForumTopic (class in
CommandObject (class in aiogram.filters.command), 580	aiogram.methods.create_forum_topic), 399
COMMANDS (aiogram.enums.menu_button_type.MenuButton	
attribute), 557	aiogram.methods.create_invoice_link), 533
commands (aiogram.methods.set_my_commands.SetMyCom	· · · · · · · · · · · · · · · · · · ·
attribute), 494	aiogram.methods.create_new_sticker_set),
CONNECTED_WEBSITE (aiogram.enums.content_type.Content_	
attribute), 550	creates_join_request

```
(aiogram.methods.create chat invite link.CreateChatInvitelitribute), 107
                                           attribute), 396
                                                                                                                                                                                                                                                                    CustomEmoji (class in aiogram.utils.formatting), 667
creates_join_request
                                                                                                                                                                                                                                                                    CZK (aiogram.enums.currency.Currency attribute), 552
                                            (aiogram.methods.edit_chat_invite_link.EditChatInviteLink
                                           attribute), 407
creates_join_request
                                                                                                                                                                                                                                                                    dark_theme_dimming(aiogram.types.background_type_fill.BackgroundTy
                                           (aiogram.types.chat invite link.ChatInviteLink
                                                                                                                                                                                                                                                                                                               attribute), 24
                                           attribute), 61
                                                                                                                                                                                                                                                                    dark_theme_dimming(aiogram.types.background_type_wallpaper.Backgr
{\tt CREATOR}\,(aiogram.enums.chat\_member\_status.ChatMemberStatus
                                                                                                                                                                                                                                                                                                              attribute), 25
                                           attribute), 548
                                                                                                                                                                                                                                                                    DART (aiogram.enums.dice emoji.DiceEmoji attribute),
creator (aiogram.types.chat_invite_link.ChatInviteLink
                                           attribute), 61
                                                                                                                                                                                                                                                                   DART (aiogram.types.dice.DiceEmoji attribute), 133
{\tt credentials} \ (a iogram.types.passport\_data. PassportData\ {\tt DATA}\ (a iogram.enums.passport\_element\_error\_type. PassportElementError\_type. PassportE
                                           attribute), 351
                                                                                                                                                                                                                                                                                                               attribute), 559
currency (a iogram. methods. create\_invoice\_link. CreateInvoice\_link. CreateInvoice\_
                                           attribute), 533
                                                                                                                                                                                                                                                                                                               tribute), 35
currency (aiogram.methods.send_invoice.SendInvoice
                                                                                                                                                                                                                                                                    data(aiogram.types.encrypted_credentials.EncryptedCredentials
                                           attribute), 538
                                                                                                                                                                                                                                                                                                               attribute), 349
currency (a iogram.types.input\_invoice\_message\_content. In put Invalige Messeges Content by the d\_passport\_element. Encrypted Passport Element in the description of the description o
                                           attribute), 328
                                                                                                                                                                                                                                                                                                               attribute), 350
currency (aiogram.types.invoice.Invoice attribute), 337
                                                                                                                                                                                                                                                                   data
                                                                                                                                                                                                                                                                                                   (aiogram.types.passport_data.PassportData
currency (aiogram.types.pre_checkout_query.PreCheckoutQuery
                                                                                                                                                                                                                                                                                                               tribute), 351
                                            attribute), 339
                                                                                                                                                                                                                                                                    data
                                                                                                                                                                                                                                                                                                     (aiogram.types.web_app_data.WebAppData
currency (aiogram.types.refunded payment.RefundedPayment
                                                                                                                                                                                                                                                                                                               tribute), 283
                                           attribute), 340
                                                                                                                                                                                                                                                                    data_hash(aiogram.types.passport_element_error_data_field.PassportEle
currency (aiogram.types.successful\_payment.SuccessfulPayment
                                                                                                                                                                                                                                                                                                               attribute), 353
                                           attribute), 346
                                                                                                                                                                                                                                                                   date(aiogram.types.business connection.BusinessConnection
Currency (class in aiogram.enums.currency), 551
                                                                                                                                                                                                                                                                                                               attribute), 32
{\tt CUSTOM\_EMOJI}\ (aiogram.enums.message\_entity\_type.Message\_\textit{Entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_\textit{Entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_type.Message\_entity\_entity\_entity\_entity\_entity\_entity\_entity\_entity\_entity\_entity\_entity\_entity\_entity\_entity\_entity\_entity\_entity\_entity\_entity\_entity\_entity\_entity\_entit
                                           attribute), 558
                                                                                                                                                                                                                                                                                                               attribute), 63
\textbf{CUSTOM\_EMOJI} \ (a iogram.enums.reaction\_type\_type. Reaction\_type\_type. Reaction\_type \\ action\_type\_type. Reaction\_types. chat\_member\_updated. ChatMemberUpdated \\ action\_type\_type. \\
                                            attribute), 560
                                                                                                                                                                                                                                                                                                               attribute), 110
CUSTOM_EMOJI (aiogram.enums.sticker_type.StickerType
                                                                                                                                                                                                                                                                   date(aiogram.types.inaccessible message.InaccessibleMessage
                                           attribute), 561
                                                                                                                                                                                                                                                                                                               attribute), 143
\verb|custom_emoji_id| (a iogram.methods.set\_custom\_emoji\_sticker_set_objumbny plesSet & \textit{sustom_emoji}\_sticker_set_objumbnail (a iogram.methods.set\_custom\_emoji\_sticker_set_objumbny plesSet_objumbny plesSet_obj
                                           attribute), 375
                                                                                                                                                                                                                                                                    \verb"date" (a iogram.types.message\_origin\_channel. MessageOriginChannel
custom_emoji_id(aiogram.types.message entity.MessageEntity
                                                                                                                                                                                                                                                                                                               attribute), 259
                                           attribute), 258
                                                                                                                                                                                                                                                                    date(aiogram.types.message_origin_chat.MessageOriginChat
 custom_emoji_id(aiogram.types.reaction_type_custom_emoji.Reaction_type_CustomEmoji
                                           attribute), 269
                                                                                                                                                                                                                                                                    \verb+date+ (a iogram.types.message\_origin\_hidden\_user. MessageOriginHiddenU
custom_emoji_id (aiogram.types.sticker.Sticker
                                                                                                                                                                                                                                                                                                               attribute), 260
                                           tribute), 335
                                                                                                                                                                                                                                                                   date(aiogram.types.message_origin_user.MessageOriginUser
custom_emoji_ids(aiogram.methods.get custom emoji
                                                                                                                                                                                                                                                                 stickers.GetGustaneEmejiStickers
                                           attribute), 370
                                                                                                                                                                                                                                                                    date(aiogram.types.message reaction count updated.MessageReactionCo
custom_emoji_sticker_set_name
                                                                                                                                                                                                                                                                                                               attribute), 261
                                            (aiogram.types.chat.Chat attribute), 37
                                                                                                                                                                                                                                                                    date (aiogram.types.message_reaction_updated.MessageReactionUpdated
custom_emoji_sticker_set_name
                                                                                                                                                                                                                                                                                                               attribute), 262
                                           (aiogram.types.chat_full_info.ChatFullInfo
                                                                                                                                                                                                                                                                    date(aiogram.types.revenue_withdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_st
                                           attribute), 61
                                                                                                                                                                                                                                                                                                               attribute), 342
custom_title (aiogram.methods.set_chat_administrator_custom_titlerant.Sypes.shat_nistratacfibnt.smafthansaction at-
                                           attribute), 484
                                                                                                                                                                                                                                                                                                               tribute), 345
custom_title(aiogram.types.chat_member_administrator@hatMgghan.typein.bitridate.Birthdate attribute), 25
                                           attribute), 105
                                                                                                                                                                                                                                                                    decline() (aiogram.types.chat_join_request.ChatJoinRequest
 custom_title(aiogram.types.chat_member_owner.ChatMemberOwner.hod), 63
```

DeclineChatJoinRequest aiogram.methods.dec 400	(class line_chat_join_red		DeleteWebhook (class in aiogram.methods.delete_webhook), 540 description(aiogram.methods.create_invoice_link.CreateInvoiceLink
decode_payload()  aiogram.utils.deep_li	(in nking), 673	module	attribute), 533 description(aiogram.methods.send_invoice.SendInvoice
DEFAULT (aiogram.enums.bot_c attribute), 547		type.BotC	
DEFAULT (aiogram.enums.menu	ı button type.Mei	nuButtonT	
attribute), 557			description(aiogram.methods.set_my_description.SetMyDescription
DefaultBotProperties	(class	in	attribute), 496
aiogram.client.defaul	*		description(aiogram.types.bot_command.BotCommand
DefaultKeyBuilder	(class	in	attribute), 26
aiogram.fsm.storage.l	base), 601		description(aiogram.types.bot_description.BotDescription
delete() (aiogram.types.me		nethod),	attribute), 30
253		,	description (aiogram.types.chat.Chat attribute), 38
DELETE_CHAT_PHOTO (aiogram	n.enums.content_ty	ype.Conte	endEspeription(aiogram.types.chat_full_info.ChatFullInfo
attribute), 550			attribute), 60
delete_chat_photo (aiogra	m.types.message.l	Message	description (aiogram.types.game.Game attribute), 364
attribute), 207	ogram.types.sticke		description(aiogram.types.inline_query_result_article.InlineQueryResult_attribute), 288
<pre>method), 335 delete_message() (aiogram</pre>			<pre>description(aiogram.types.inline_query_result_cached_document.Inlin</pre>
41		,,	description(aiogram.types.inline_query_result_cached_photo.InlineQu
delete_photo() (aiogram.typ	es.chat.Chat meth	nod), 50	attribute), 300
delete_reply_markup()			description(aiogram.types.inline_query_result_cached_video.InlineQu
(aiogram.types.messa	ige.Message r	nethod),	attribute), 304
250			description(aiogram.types.inline_query_result_document.InlineQueryI
delete_sticker_set()	(aiogram.types.ch	hat.Chat	attribute), 310
method), 44			${\tt description} (\it aiogram.types.inline\_query\_result\_photo.InlineQueryResult$
DeleteChatPhoto	(class	in	attribute), 319
aiogram.methods.dele	ete_chat_photo), 4	-02	${\tt description} ({\it aiogram.types.inline\_query\_result\_video.Inline} {\it QueryResult\_video.Inline} {\it Q$
DeleteChatStickerSet	(class	in	attribute), 323
aiogram.methods.dele 403	ete_chat_sticker_s	et),	description (aiogram.types.input_invoice_message_content.InputInvoice_attribute), 328
DELETED_BUSINESS_MESSAGE	S		description (aiogram.types.invoice.Invoice attribute),
(aiogram.enums.upda	te_type.UpdateTy	pe	337
attribute), 562			deserialize_telegram_object() (in module
deleted_business_message			aiogram.utils.serialization), 674
(aiogram.types.updat	e.Update at	ttribute),	
361			module aiogram.utils.serialization), 674
DeleteForumTopic	(class		5 5 \
aiogram.methods.dele			aiogram.utils.serialization), 674
DeleteMessage	(class		DetailedAiogramError, 626
aiogram.methods.dele			DICE (aiogram.enums.content_type.ContentType at-
DeleteMessages	(class	in	tribute), 549
aiogram.methods.dele			DICE (aiogram.enums.dice_emoji.DiceEmoji attribute),
DeleteMyCommands	(class	in	554
aiogram.methods.dele	ete_my_commands	s),	DICE (aiogram.types.dice.DiceEmoji attribute), 133
405	. 1		dice (aiogram.types.external_reply_info.ExternalReplyInfo
DeleteStickerFromSet	(class	in	attribute), 135
aiogram.methods.dele	ete_sticker_from_s	set),	dice (aiogram.types.message.Message attribute), 207
368	(-1		Dice (class in aiogram.types.dice), 133
DeleteStickerSet	(class		DiceEmoji (class in aiogram.enums.dice_emoji), 554
aiogram.methods.dele	eie_sticker_set), 3	09	DiceEmoji (class in aiogram.types.dice), 133

disable() (aiogram.utils.callback_answer.CallbackAnswe	er	(aiogram.methods.send_media_group.SendMediaGroup
method), 661		attribute), 460
disable_content_type_detection		_notification
(aiogram.methods.send_document.SendDocumen attribute), 455	et e	(aiogram.methods.send_message.SendMessage attribute), 462
disable_content_type_detection	disable	_notification
		ehtiogram.methods.send_paid_media.SendPaidMedia
attribute), 189		attribute), 465
disable_edit_message	disable	_notification
(aiogram.methods.set_game_score.SetGameScore		(aiogram.methods.send_photo.SendPhoto
attribute), 529		attribute), 468
disable_notification	disable	_notification
(aiogram.client.default.DefaultBotProperties		(aiogram.methods.send_poll.SendPoll at-
attribute), 567		tribute), 471
disable_notification	disable	_notification
(aiogram.methods.copy_message.CopyMessage		(aiogram.methods.send_sticker.SendSticker
attribute), 393		attribute), 374
disable_notification	disable	_notification
(aiogram.methods.copy_messages.CopyMessages	S	(aiogram.methods.send_venue.SendVenue
attribute), 395		attribute), 474
disable_notification	disable	_notification
(aiogram.methods.forward_message.ForwardMes	ssage	(aiogram.methods.send_video.SendVideo
attribute), 413		attribute), 477
disable_notification	disable	_notification
(aiogram.methods.forward_messages.ForwardMe	essages	(aiogram.methods.send_video_note.SendVideoNote
attribute), 414		attribute), 480
disable_notification	disable	_notification
(aiogram.methods.pin_chat_message.PinChatMe	ssage	(aiogram.methods.send_voice.SendVoice
attribute), 434		attribute), 482
disable_notification		_web_page_preview
$(aiogram.methods.send\_animation.Send Animation) \\$	n	(aiogram.methods.edit_message_text.EditMessageText
attribute), 443		attribute), 518
disable_notification	disable	_web_page_preview
(aiogram.methods.send_audio.SendAudio		(aiogram.methods.send_message.SendMessage
attribute), 446		attribute), 463
disable_notification	disable	_web_page_preview
(aiogram.methods.send_contact.SendContact		(aiogram.types.input_text_message_content.InputTextMessageC
attribute), 450	1. 1.	attribute), 331
disable_notification	disable	d (aiogram.utils.callback_answer.CallbackAnswer
(aiogram.methods.send_dice.SendDice at-	D	property), 661
tribute), 452	Dispato	her (class in aiogram.dispatcher.dispatcher),
disable_notification		575
(atogram.methoas.sena_aocument.senaDocumen attribute), 455	<i>u</i> aistanc	e (aiogram.types.proximity_alert_triggered.ProximityAlertTrigge attribute), 267
disable_notification	DKK (aios	gram.enums.currency.Currency attribute), 552
(aiogram.methods.send_game.SendGame		ogram.types.chat.Chat method), 43
attribute), 527		T (aiogram.enums.content_type.ContentType at-
disable_notification		tribute), 549
(aiogram.methods.send_invoice.SendInvoice	DOCUMEN	T (aiogram.enums.inline_query_result_type.InlineQueryResultTy
attribute), 539		attribute), 555
disable_notification	DOCUMEN	T (aiogram.enums.input_media_type.InputMediaType
$(a iogram. methods. send\_location. Send Location$		attribute), 556
attribute), 458	documen	t (aiogram.methods.send_document.SendDocument

attribute), 454

disable\_notification

document (aiogram.types.background_type_pattern.Backgr	andTypePattern	ı			
attribute), 24	edit_caption(	() (aiog	gram.types.messag	e.Message	
${\tt document} \ (a iogram. types. background\_type\_wall paper. Background\_type\_wall pa$	groundTypeWali	lpaper2	71	0	
attribute), 25	edit_date (aid	ogram.types.n	nessage.Message	attribute),	
document (aiogram.types.external_reply_info.ExternalRep	200				
attribute), 135	edit_invite_]	link()	(aiogram.types	.chat.Chat	
document (aiogram.types.message.Message attribute),	method	* *			
206	edit_live_lo				
Document (class in aiogram.types.document), 133 document_file_id(aiogram.types.inline_query_result_c	(aiogra chad document	am.types.mes	sage.Message	method),	
attribute), 294	chea_aocusteni.	immeQueryr	<i>Resuired Cheapoet</i>	anteni M	
document url (aiogram types inline query result docum	edit_media() ent.InlineOuervR	aiog) ResultDocume	ram.types.messag ent	e.Message	
<pre>attribute), 294 document_url (aiogram.types.inline_query_result_docum     attribute), 310</pre>	edit_reply_ma	a), 249			
DOP (aiogram.enums.currency.Currency attribute), 552	carc_rcpry_me	ur Kup ()	sage.Message	method),	
download() (aiogram.client.bot.Bot method), 564	250	am.iypes.mes	suge.messuge	memoa),	
		aiooram tynes	s message Messag	e method)	
down1oad_±11e() (aiogram.client.bot.Bot method), 563 DRIVER_LICENSE (aiogram.enums.encrypted_passport_ele	nent.EncryptedF	PassportElem	ent	c memou),	
attribute), 554	EditChatInvi		(class	in	
drop_pending_updates	aiogra		dit_chat_invite_lir		
(aiogram.methods.delete_webhook.DeleteWebhoo	k 406			,,	
attribute), 540	EditChatSubso	criptionInv	viteLink (cl	lass in	
drop_pending_updates	aiogra	ım.methods.ed	dit_chat_subscrip	tion_invite_i	link),
(aiogram.methods.set_webhook.SetWebhook	408				
attribute), 544	EDITED_BUSIN	ESS_MESSAGI	Ε		
duration (aiogram.methods.send_animation.SendAnimati	(utogri		date_type.Update	Type	
attribute), 442 duration (aiogram.methods.send_audio.SendAudio at-		ute), 562			
tribute), 446	edited_busine	_			
duration (aiogram.methods.send_video.SendVideo at-		am.types.upd	ate.Update	attribute),	
tribute), 476	361	EL DOCT			
duration (aiogram.methods.send_video_note.SendVideoN	EDITED_CHANNI		data tuna Undata	Tun a	
attribute), 479	(ulogn	am.enums.up ute), 562	date_type.Update	туре	
duration (aiogram.methods.send_voice.SendVoice at-	edited_channe		iogram.types.updo	ate Undate	
tribute), 482		ute), 361	iogram.iypes.upac	ле.Орише	
duration (aiogram.types.animation.Animation at-		* *	enums.update_type	e.UndateTvn	e.
tribute), 19		ute), 562	perez.jp	or operation typ	
duration (aiogram.types.audio.Audio attribute), 20	aditad massac	an (ainara)	m.types.update.Up	date at-	
${\tt duration} \ (a iogram. types. input\_media\_animation. Input Media\_animation. Input Medi$	liaAnimationbute	2), 361			
attribute), 187	EditForumTop	ic	(class	in	
duration(aiogram.types.input_media_audio.InputMediaA	udio aiogra	ım.methods.ed	dit_forum_topic),	409	
attribute), 188	EditGeneralFo	orumTopic	(class	in	
duration(aiogram.types.input_media_video.InputMediaV	<sup>deo</sup> aiogra	ım.methods.ed	dit_general_forum	ı_topic),	
attribute), 191	410				
duration(aiogram.types.input_paid_media_video.InputPo attribute), 193			(class	in	
duration(aiogram.types.paid_media_preview.PaidMedia.		ım.methods.ed	dit_message_capti	ion),	
attribute), 264	307				
duration (aiogram.types.video.Video attribute), 280	EditMessageLi			in	
duration(aiogram.types.video_chat_ended.VideoChatEnd	ed 511	ım.meinoas.ed	dit_message_live_	<u>iocanon),</u>	
attribute), 280	EditMessageMe	edia	(class	in	
duration (aiogram.types.video_note.VideoNote at-	_		(ciass dit_message_medi		
tribute), 282	aiogra EditMessageRe		an_message_mean (class	ia), 313 in	
duration (aiogram.types.voice.Voice attribute), 282	_		dit_message_reply		
DZD (aiogram.enums.currency.Currency attribute), 552	515		80_,601)	,	

EditMessageText	(class is	ı	attribute), 462		
aiogram.methods.edii	•		· · · · · · · · · · · · · · · · · · ·	_text_message_con	tent.InputTextMessageCo
effect_id (aiogram.types.me			attribute), 331		
206		entitie	s (aiogram.types.mes.	sage.Message attr	ibute),
EGP (aiogram.enums.currency.	Currency attribute), 552		206		
element_hash(aiogram.types.attribute), 359	.passport_element_error_	_uns <mark>qnet ifteid</mark> e.	Bassp( <b>ovitIghame.ty/Ees:10e</b> ) tribute), 275	kt <u>n</u> spætifiFelxtQuote	at-
EMAIL (aiogram.enums.encrypt attribute), 555	ed_passport_element.End	cryp <b>eachBassp</b>	<b>essEdgæ¢at</b> ogram.meth attribute), 530	ods.answer_pre_ch	eckout_query.AnswerPr
EMAIL (aiogram.enums.message attribute), 557	e_entity_type.MessageEn	<i>tity<b>E</b>yppe</i> or_m	**	ods.answer_shippir	ng_query.AnswerShippir
email(aiogram.types.encrypte	d_passport_element.Enci	ypt <b>EddPoorsFoo</b>	**	types.error_event),	625
attribute), 350		errors(	aiogram.methods.set_p	passport_data_erro	rs.SetPassportDataErroi
email (aiogram.types.order_a	info.OrderInfo attribute)		attribute), 546		
338			gram.enums.currency.C		
Email (class in aiogram.utils.fe	_		gram.enums.currency.C		
EMOJI (aiogram.enums.reaction	n_type_type.ReactionType				
attribute), 560			ype (aiogram.types.u	pdate.Update prop	perty),
emoji (aiogram.methods.send	_dice.SendDice attribute)		362		
452			on (aiogram.types.err	ror_event.ErrorEvei	nt at-
	ls.send_sticker.SendSticke		<i>tribute</i> ), 625		
attribute), 373		Excepti	onMessageFilter	(class	in
emoji (aiogram.types.dice.Dic			aiogram.filters.excepti		Til.
emoji (aiogram.types.reaction_attribute), 269		-	attribute), 589	ception.ExceptionTy	vpeFilter
emoji (aiogram.types.sticker.S		_	onTypeFilter	(class	in
emoji_list(aiogram.methods	s.set_sticker_emoji_list.Se				
attribute), 376			aiogram.fsm.scene.Sce	neWizard method),	620
	s.input_sticker.InputSticke	r EXPANDA	BLE_BLOCKQUOTE		
attribute), 333			(aiogram.enums.messo	age_entity_type.Me	ssageEntityType
emoji_status_custom_emoj			attribute), 558		
(aiogram.types.chat.C		Expanda	bleBlockQuote	(class	in
emoji_status_custom_emoj			aiogram.utils.formatti		. <b>n</b>
	full_info.ChatFullInfo	expirat	ion_date(aiogram.typ	pes.chat_boost.Cha	tBoost
attribute), 59	4-4-		attribute), 54	l	link Consul Chartenite
<pre>emoji_status_expiration_</pre>		expire_		s.create_cnat_invite	e_link.CreateChatInvite
emoji_status_expiration_	, ,	ovniro	attribute), 396	ls adit chat invita	ink.EditChatInviteLink
	full_info.ChatFullInfo	expire_	attribute), 407	s.eatt_chat_thvtte_t	ink.EanChaimviieEink
attribute), 60	,uu_ugo.Chan uungo	evnire	date( <i>aiogram.types.cl</i>	hat invite link Cha	tInviteI ink
EncryptedCredentials	(class i	n expire_	attribute), 62	iai_inviie_iink.Cna	imviieLink
	oted_credentials), 349		tion (aiogram.method	ds send noll SendPo	oll at-
EncryptedPassportElement		п скртина	<i>tribute</i> ), 471	is.sena_pon.senar	ii di
	vpted_passport_element),		tion (aiogram.types.pe	oll.Poll attribute). 2	266
554	<i></i>	_	tion_entities		
EncryptedPassportElement	: (class i	_	(aiogram.methods.sen	d poll.SendPoll	at-
	oted_passport_element),		<i>tribute</i> ), 471	·· <b>_</b> r · · · · · · · · · · · · · · · · · · ·	
350	<b>-1</b> 1 - //	explana	tion_entities	(aiogram.types.po	ll.Poll
enter() (aiogram.fsm.scene	ScenesManager method	-	attribute), 266	. 0 11 1	
618	,		tion_parse_mode		
<pre>enter() (aiogram.fsm.scene.S</pre>	ceneWizard method), 619	_	(aiogram.methods.sen	d_poll.SendPoll	at-
entities (aiogram.methods.ed			tribute), 471		
attribute), 517		export(	) (aiogram.utils.keyboo	ard.InlineKeyboard	Builder
entities (aiogram.methods.se	end_message.SendMessag	re	method), 645		

<pre>export() (aiogram.utils.keyboard.ReplyKeyboardBuilder</pre>	<pre>file_id (aiogram.types.document.Document attribute),</pre>
<pre>export_invite_link() (aiogram.types.chat.Chat</pre>	file_id (aiogram.types.file.File attribute), 136
method), 43	file_id (aiogram.types.passport_file.PassportFile at- tribute), 360
<del>-</del>	
<pre>aiogram.methods.export_chat_invite_link), 411</pre>	file_id (aiogram.types.photo_size.PhotoSize attribute), 265
external_reply (aiogram.types.message.Message attribute), 205	file_id (aiogram.types.sticker.Sticker attribute), 334 file_id (aiogram.types.video.Video attribute), 279
ExternalReplyInfo (class in aiogram.types.external_reply_info), 134	file_id (aiogram.types.video_note.VideoNote at- tribute), 281
extract_flags() (in module aiogram.dispatcher.flags),	file_id (aiogram.types.voice.Voice attribute), 282
628	file_name (aiogram.types.animation.Animation at-
extract_from() (aiogram.types.message_entity.Message.	
method), 258	file_name (aiogram.types.audio.Audio attribute), 20
EYES (aiogram.enums.mask_position_point.MaskPositionPoint.maskPoint.maskPositionPoint.maskPositionPoint.maskPositionPoint.maskPoint.maskPositionPoint.maskPositionPoint.maskPositionPoint.maskPo	offitle_name (aiogram.types.document.Document at- tribute), 134
anticonic), ee,	file_name (aiogram.types.video.Video attribute), 280
F	file_path (aiogram.types.file.File attribute), 136
FAILED (aiogram.enums.revenue_withdrawal_state_type.Reattribute), 560	e <del>\$\delta intermalisation.</del> Animation at- tribute), 20
feed_raw_update() (aiogram.dispatcher.dispatcher.Disp	afille_size (aiogram.types.audio.Audio attribute), 21
method), 575	file_size (aiogram.types.document.Document at-
${\tt feed\_update()}\ (a iogram. dispatcher. Dispatcher$	r tribute), 134
method), 576	file_size (aiogram.types.file.File attribute), 136
field_name (aiogram.types.passport_element_error_data attribute), 353	tribute), 360
file (aiogram.client.telegram.TelegramAPIServer attribute), 14	file_size (aiogram.types.photo_size.PhotoSize at- tribute), 265
FILE (aiogram.enums.passport_element_error_type.Passport_element_element_error_type.Pa	fielementspeam.types.sticker.Sticker attribute), 335 file_size (aiogram.types.video.Video attribute), 280
File (class in aiogram.types.file), 136	<pre>file_size (aiogram.types.video_note.VideoNote at-</pre>
file_date (aiogram.types.passport_file.PassportFile at-	tribute), 282
tribute), 360	file_size (aiogram.types.voice.Voice attribute), 282
file_hash(aiogram.types.passport_element_error_file.Paattribute), 353	stile English (Ar Figiogram.types.animation.Animation attribute), 19
file_hash(aiogram.types.passport_element_error_front_attribute), 355	stal Passpinweleink (ainstarpromesaudio Audio attribute), 20
file_hash(aiogram.types.passport_element_error_revers attribute), 356	efile. Paisportideme aiosrane vorse document. Document attribute), 133
file_hash(aiogram.types.passport_element_error_selfie. attribute), 357	
file_hash(aiogram.types.passport_element_error_transl attribute), 358	
file_hashes (aiogram.types.passport_element_error_file.	
attribute), 354	file_unique_id (aiogram.types.sticker.Sticker at-
file_hashes (aiogram.types.passport_element_error_tranatribute), 359	nslation_files.PetssportElementErrorTranslationFiles file_unique_id (aiogram.types.video.Video attribute),
file_id (aiogram.methods.get_file.GetFile attribute), 421	279 file_unique_id (aiogram.types.video_note.VideoNote
file_id (aiogram.types.animation.Animation attribute),	attribute), 281
19	file_unique_id (aiogram.types.voice.Voice attribute),
file_id (aiogram.types.audio.Audio attribute), 20	282
	file_url() (aiogram.client.telegram.TelegramAPIServer

method), 14	FORUM_TOPIC_CREATED
FILES (aiogram.enums.passport_element_error_type.Passp	
attribute), 559	attribute), 550
${\tt files} \ (a iogram. types. encrypted\_passport\_element. Encrypted\_element. Encrypted\_passport\_element. Encrypted\_passport\_element. Encrypted\_passport\_element. Encrypted\_passport\_element. Encrypted\_passport\_element. Encrypted\_passport\_element. Encrypted\_passport\_element. Encrypted\_passport\_element. Encrypt$	-
attribute), 351	(aiogram.types.message.Message attribute),
fill (aiogram.types.background_type_fill.BackgroundType attribute), 24	Fill 209  FORUM_TOPIC_EDITED (aiogram.enums.content_type.ContentType
fill (aiogram.types.background_type_pattern.Background	
attribute), 24	forum_topic_edited(aiogram.types.message.Message
Filter (class in aiogram.filters.base), 590	attribute), 209
<pre>filter() (aiogram.filters.callback_data.CallbackData</pre>	
class method), 587	(aiogram.enums.content_type.ContentType
FIND_LOCATION (aiogram.enums.chat_action.ChatAction	attribute), 550
attribute), 548	forum_topic_reopened
find_location() (aiogram.utils.chat_action.ChatActionSclass method), 652	209
${\tt first\_name} \ (a iogram.methods.send\_contact.SendContact$	
attribute), 449	ForumTopicClosed (class in
first_name (aiogram.types.chat.Chat attribute), 36	aiogram.types.forum_topic_closed), 138
first_name (aiogram.types.chat_full_info.ChatFullInfo attribute), 59	ForumTopicCreated (class in
first_name (aiogram.types.contact.Contact attribute),	aiogram.types.forum_topic_created), 138 ForumTopicEdited (class in
132	aiogram.types.forum_topic_edited), 139
first_name (aiogram.types.inline_query_result_contact.In	
attribute), 307	aiogram.types.forum_topic_reopened), 139
<pre>first_name (aiogram.types.input_contact_message_conte</pre>	
attribute), 326	249
<pre>first_name (aiogram.types.shared_user.SharedUser at- tribute), 273</pre>	forward_date (aiogram.types.message.Message at- tribute), 210
<pre>first_name (aiogram.types.user.User attribute), 275</pre>	forward_from (aiogram.types.message.Message at-
	tribute), 210
attribute), 656	forward_from_chat (aiogram.types.message.Message
FOOTBALL (aiogram.enums.dice_emoji.DiceEmoji	attribute), 210
attribute), 554	forward_from_message_id
FOOTBALL (aiogram.types.dice.DiceEmoji attribute), 133 for_channels (aiogram.methods.get_my_default_adminis	(aiogram.types.message.Message attribute), trator_rights.GetMyDefaultAdministratorRights
attribute), 425	forward_origin (aiogram.types.message.Message at-
for_channels (aiogram.methods.set_my_default_adminis	
attribute), 495	forward_sender_name
force (aiogram.methods.set_game_score.SetGameScore attribute), 529	(aiogram.types.message.Message attribute), 210
force_reply (aiogram.types.force_reply.ForceReply attribute), 137	forward_signature (aiogram.types.message.Message attribute), 210
ForceReply (class in aiogram.types.force_reply), 137	forward_text (aiogram.types.login_url.LoginUrl
FOREHEAD (aiogram.enums.mask_position_point.MaskPos	
attribute), 557	ForwardMessage (class in
format (aiogram.methods.set_sticker_set_thumbnail.SetSti	· · · · · · · · · · · · · · · · · · ·
attribute), 381  format (aiogram.types.input sticker.InputSticker at-	ForwardMessages (class in aiogram.methods.forward_messages), 414
format (aiogram.types.input_sticker.InputSticker at- tribute), 333	foursquare_id (aiogram.methods.send_venue.SendVenue
FORUM_TOPIC_CLOSED (aiogram.enums.content_type.Cont	
attribute), 550	foursquare_id(aiogram.types.inline_query_result_venue.InlineQueryRe.
forum_topic_closed (aiogram.types.message.Message	attribute), 320
attribute), 209	foursquare_id(aiogram.types.input_venue_message_content.InputVenue

```
attribute), 332
                                                                                         from_user(aiogram.types.pre_checkout_query.PreCheckoutQuery
foursquare_id (aiogram.types.venue.Venue attribute),
                                                                                                        attribute), 339
                                                                                         from_user(aiogram.types.shipping_query.ShippingQuery
foursquare_type (aiogram.methods.send_venue.SendVenue
                                                                                                        attribute), 344
              attribute), 473
                                                                                         FRONT_SIDE (aiogram.enums.passport_element_error_type.PassportEleme
foursquare_type (aiogram.types.inline query result venue.Inline QueripRies) ul Wenue
              attribute), 321
                                                                                         front_side(aiogram.types.encrypted passport element.EncryptedPasspo
foursquare_type (aiogram.types.input_venue_message_content.InpathibuteMessageContent
              attribute), 332
                                                                                         FSInputFile (class in aiogram.types.input file), 185,
foursquare_type
                                                                                                        565
                                  (aiogram.types.venue.Venue
              tribute), 279
                                                                                        FSMI18nMiddleware
                                                                                                                                              (class
                                                                                                                                                                           in
FRAGMENT (aiogram.enums.transaction_partner_type.TransactionPartner_type.utils.i18n.middleware), 649
                                                                                        FSMStrategy (class in aiogram.fsm.strategy), 603
              attribute), 561
from_attachment_menu
                                                                                         full_name (aiogram.types.chat.Chat property), 40
              (aiogram.types.write_access_allowed.WriteAccessAllowadame (aiogram.types.user.User property), 276
              attribute), 284
{\tt from\_base()} \ ({\it aiogram.client.telegram.TelegramAPIServer} G
               class method), 15
                                                                                         GAME
                                                                                                      (aiogram.enums.content_type.ContentType
from_chat_id(aiogram.methods.copy message.CopyMessage
                                                                                                        tribute), 549
               attribute), 392
                                                                                         GAME (aiogram.enums.inline_query_result_type.InlineQueryResultType
from_chat_id(aiogram.methods.copy_messages.CopyMessages
                                                                                                        attribute), 555
              attribute), 395
                                                                                         game (aiogram.types.external_reply_info.ExternalReplyInfo
from_chat_id(aiogram.methods.forward_message.ForwardMessageattribute), 136
               attribute), 412
                                                                                         game (aiogram.types.message.Message attribute), 207
{\tt from\_chat\_id} \ (a iogram. methods. forward\_messages. Forward\_messages. Forward\_messages in a iogram. types. game), 364
              attribute), 414
                                                                                         game_short_name (aiogram.methods.send_game.SendGame
from_file() (aiogram.types.input_file.BufferedInputFile
                                                                                                        attribute), 527
              class method), 184
                                                                                         game_short_name(aiogram.types.callback query.CallbackQuery
from_markup() (aiogram.utils.keyboard.InlineKeyboardBuilder
                                                                                                        attribute), 35
              class method), 645
                                                                                         game_short_name(aiogram.types.inline_query_result_game.InlineQueryI
from_markup() (aiogram.utils.keyboard.ReplyKeyboardBuilder
                                                                                                        attribute), 311
              class method), 646
                                                                                         GameHighScore
                                                                                                                                          (class
                                                                                                                                                                           in
from\_request ({\it aiogram.types.write\_access\_allowed.WriteAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessA\bar{l}lowed.writeAccessAccessA\bar{l}lowed.writeAccessAccessAccessAccessAccessAccessAccessAccessAccessAccessAccessAccessAccessAccessAccessAccessAccessAccessAccessAccessAccessAccessAccessAccessAccessAccessAccessAccessAccessAcc
              attribute), 283
                                                                                        GBP (aiogram.enums.currency.Currency attribute), 552
from_url() (aiogram.fsm.storage.mongo.MongoStorage
                                                                                        GEL (aiogram.enums.currency.Currency attribute), 552
              class method), 601
                                                                                         GENERAL_FORUM_TOPIC_HIDDEN
from_url()
                         (aiogram.fsm.storage.redis.RedisStorage
                                                                                                        (aiogram.enums.content_type.ContentType
              class method), 600
                                                                                                        attribute), 550
from_user(aiogram.handlers.callback_query.CallbackQueryHerreHerforum_topic_hidden
              property), 639
                                                                                                        (aiogram.types.message.Message
                                                                                                                                                               attribute),
from_user(aiogram.types.callback query.CallbackQuery
               attribute), 34
                                                                                        GENERAL_FORUM_TOPIC_UNHIDDEN
from_user(aiogram.types.chat join request.ChatJoinRequest
                                                                                                        (aiogram.enums.content_type.ContentType
              attribute), 62
                                                                                                        attribute), 550
from\_user({\it aiogram.types.chat\_member\_updated.ChatMember\_lpdated} rum\_topic\_unhidden
              attribute), 110
                                                                                                        (aiogram.types.message.Message
                                                                                                                                                               attribute),
from_user(aiogram.types.chosen_inline_result.ChosenInlineResult 209
              attribute), 284
                                                                                         GeneralForumTopicHidden
                                                                                                                                                    (class
                                                                                                                                                                           in
from_user (aiogram.types.inline_query.InlineQuery at-
                                                                                                        aiogram.types.general_forum_topic_hidden),
               tribute), 285
                                                                                                        139
from_user (aiogram.types.message.Message attribute),
                                                                                        GeneralForumTopicUnhidden
                                                                                                                                                      (class
                                                                                                                                                                           in
                                                                                                        aiogram.types.general_forum_topic_unhidden),
from_user(aiogram.types.paid_media_purchased.PaidMediaPurchased)
              attribute), 338
                                                                                         get() (aiogram.fsm.scene.SceneRegistry method), 618
```

<pre>get_administrators()     method), 41</pre>	(aiogram.types.chat.C	hat	aiogram.methods.get_my_short_description), 428
<pre>get_data() (aiogram.fsm.scc</pre>	ene.SceneWizard metho	(d),	
620			aiogram.methods.get_star_transactions),
	ı.storage.base.BaseStor	age	535
method), 602	1, 1, 0, 1, 0	.0	GetStickerSet (class in
<pre>get_flag() (in module aiogra</pre>			aiogram.methods.get_sticker_set), 371
<i>method</i> ), 650			deGartUpdates (class in aiogram.methods.get_updates), 541
<pre>get_member() (aiogram.types</pre>			GetUserChatBoosts (class in
<pre>get_member_count()     method), 45</pre>	(aiogram.types.chat.C	'hat	aiogram.methods.get_user_chat_boosts), 428
<pre>get_profile_photos()     method), 276</pre>	(aiogram.types.user.U	ser	GetUserProfilePhotos (class in aiogram.methods.get_user_profile_photos),
<pre>get_state() (aiogram.fsm     method), 602</pre>	n.storage.base.BaseStor	age	429 GetWebhookInfo (class in
get_url() (aiogram.types.m	essage Message metho	od)	aiogram.methods.get_webhook_info), 543
254	essage.memo	α),	GIF (aiogram.enums.inline_query_result_type.InlineQueryResultType
GetBusinessConnection	(class	in	attribute), 555
	_business_connection),		gif_duration(aiogram.types.inline_query_result_gif.InlineQueryResult attribute), 313
GetChat (class in aiogram.me	thods get_chat) 416		gif_file_id(aiogram.types.inline_query_result_cached_gif.InlineQuer
GetChatAdministrators	(class	in	attribute), 295
	chat_administrators),		gif_height (aiogram.types.inline_query_result_gif.InlineQueryResultGi_ attribute), 313
GetChatMember	(class	in	gif_url (aiogram.types.inline_query_result_gif.InlineQueryResultGif
aiogram.methods.get	_chat_member), 418	ırı	attribute), 312
GetChatMemberCount  aiogram.methods.get	(class _chat_member_count),	in	<pre>gif_width(aiogram.types.inline_query_result_gif.InlineQueryResultGif</pre>
419			GIFT_CODE (aiogram.enums.chat_boost_source_type.ChatBoostSourceType)
GetChatMenuButton	(class	in	attribute), 548
aiogram.methods.get 420	_chat_menu_button),		GIVEAWAY (aiogram.enums.chat_boost_source_type.ChatBoostSourceType attribute), 548
GetCustomEmojiStickers	(class	in	GIVEAWAY (aiogram.enums.content_type.ContentType at-
_	_custom_emoji_stickers		tribute), 551
370		,,	${\tt giveaway}  (a iogram. types. external\_reply\_info. ExternalReplyInfo$
GetFile (class in aiogram.me			attribute), 136
GetForumTopicIconSticker			giveaway (aiogram.types.message.Message attribute),
	_forum_topic_icon_stic	kers	
422	( 1		Giveaway (class in aiogram.types.giveaway), 140
GetGameHighScores	(class	in	GIVEAWAY_COMPLETED (aiogram.enums.content_type.ContentType
525	g_game_high_scores),		attribute), 551 giveaway_completed (aiogram.types.message.Message
GetMe (class in aiogram.metho	ods.get_me), 423		attribute), 209
GetMyCommands	(class	in	GIVEAWAY_CREATED (aiogram.enums.content_type.ContentType
	_my_commands), 424		attribute), 550
GetMyDefaultAdministrato	•	in	5
· ·	_my_default_administro	ator_	
425	. 1		giveaway_message(aiogram.types.giveaway_completed.GiveawayComp
GetMyDescription	(class	in	attribute), 141
	_my_description), 426	10)	giveaway_message_id
GetMyName (class in aiogra 427	m.meinoas.gei_my_nan	<i>ie)</i> ,	(aiogram.types.chat_boost_source_giveaway.ChatBoostSourceCattribute), 56
GetMyShortDescription	(class	jn	giveaway_message_id
,	(0.0000		g = · <b></b>

```
(aiogram.types.giveaway winners.GiveawayWinners_hidden_members (aiogram.types.chat.Chat at-
                   attribute), 142
                                                                                                                                      tribute), 38
GIVEAWAY_WINNERS (aiogram.enums.content_type.ContentThans_hidden_members (aiogram.types.chat_full_info.ChatFullInfo
                                                                                                                                      attribute), 61
                   attribute), 551
giveaway_winners (aiogram.types.external_reply_info.ExharsahReinly.hnsb_app (aiogram.types.user.User attribute),
                  attribute), 136
                                                                                                                                      276
giveaway_winners
                                              (aiogram.types.message.Message has_media_spoiler(aiogram.types.external reply info.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.ExternalReplyInfo.E
                   attribute), 209
                                                                                                                                      attribute), 135
GiveawayCompleted
                                                                     (class
                                                                                                                 has_media_spoiler (aiogram.types.message.Message
                   aiogram.types.giveaway_completed), 141
                                                                                                                                      attribute), 207
GiveawayCreated
                                                                  (class
                                                                                                          in
                                                                                                                  has_private_forwards (aiogram.types.chat.Chat at-
                                                                                                                                      tribute), 38
                  aiogram.types.giveaway_created), 141
GiveawayWinners
                                                                                                                 has_private_forwards
                                                                  (class
                                                                                                          in
                  aiogram.types.giveaway_winners), 142
                                                                                                                                      (aiogram.types.chat_full_info.ChatFullInfo
GLOBAL_USER
                                          (aiogram.fsm.strategy.FSMStrategy
                                                                                                                                      attribute), 60
                   attribute), 603
                                                                                                                  has_protected_content (aiogram.types.chat.Chat at-
google_place_id(aiogram.methods.send_venue.SendVenue
                                                                                                                                      tribute), 38
                   attribute), 474
                                                                                                                  has_protected_content
google_place_id(aiogram.types.inline_query_result_venue.InlineQuaiogRaenutyVenuchat_full_info.ChatFullInfo
                   attribute), 321
                                                                                                                                      attribute), 61
google_place_id(aiogram.types.input_venue_message_chatsuphopted/tende/MonsteppeContent
                   attribute), 332
                                                                                                                                      (aiogram.types.message.Message
                                                                                                                                                                                                             attribute),
google_place_id
                                           (aiogram.types.venue.Venue
                   tribute), 279
                                                                                                                  has_public_winners (aiogram.types.giveaway.Giveaway
google_place_type (aiogram.methods.send_venue.SendVenue
                                                                                                                                      attribute), 140
                   attribute), 474
                                                                                                                  has_restricted_voice_and_video_messages
google_place_type (aiogram.types.inline_query_result_venue.Inlin@QovernyNetsypleV.chuat.Chat attribute), 38
                   attribute), 321
                                                                                                                  has_restricted_voice_and_video_messages
google_place_type (aiogram.types.input_venue_message_content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqive\textit{venue_Message_Content.ltqiv
                   attribute), 332
                                                                                                                                      attribute), 60
google_place_type
                                                        (aiogram.types.venue.Venue
                                                                                                                 has_spoiler(aiogram.methods.send_animation.SendAnimation
                   attribute), 279
                                                                                                                                      attribute), 443
goto() (aiogram.fsm.scene.SceneWizard method), 620
                                                                                                                  has_spoiler (aiogram.methods.send_photo.SendPhoto
GREEN (aiogram.enums.topic_icon_color.TopicIconColor
                                                                                                                                      attribute), 468
                   attribute), 561
                                                                                                                  has_spoiler (aiogram.methods.send video.SendVideo
GROUP (aiogram.enums.chat_type.ChatType attribute),
                                                                                                                                      attribute), 477
                                                                                                                  has_spoiler(aiogram.types.input media animation.InputMediaAnimatic
GROUP_CHAT_CREATED (aiogram.enums.content_type.ContentType
                                                                                                                                     attribute), 187
                   attribute), 550
                                                                                                                  has_spoiler(aiogram.types.input_media_photo.InputMediaPhoto
group_chat_created (aiogram.types.message.Message
                                                                                                                                      attribute), 190
                  attribute), 208
                                                                                                                  has_spoiler(aiogram.types.input media video.InputMediaVideo
GTQ (aiogram.enums.currency.Currency attribute), 552
                                                                                                                                      attribute), 191
                                                                                                                  has_visible_history
                                                                                                                                                                                (aiogram.types.chat.Chat
Н
                                                                                                                                      attribute), 38
                                                                                                                  has_visible_history
handlers (aiogram.fsm.scene.SceneConfig attribute),
                                                                                                                                      (aiogram.types.chat_full_info.ChatFullInfo
                   619
                                                                                                                                      attribute), 61
has_aggressive_anti_spam_enabled
                                                                                                                  hash(aiogram.types.encrypted_credentials.EncryptedCredentials
                   (aiogram.types.chat.Chat attribute), 38
                                                                                                                                      attribute), 349
has_aggressive_anti_spam_enabled
                                                                                                                  hash(aiogram.types.encrypted_passport_element.EncryptedPassportEleme
                   (aiogram.types.chat full info.ChatFullInfo
                                                                                                                                      attribute), 350
                   attribute), 60
                                                                                                                  hash (aiogram.utils.web_app.WebAppInitData attribute),
has_custom_certificate
                   (aiogram.types.webhook_info.WebhookInfo
                                                                                                                  HASHTAG (aiogram.enums.message entity type.MessageEntityType
                   attribute), 363
```

id(aiogram.types.inline\_query\_result\_cached\_photo.InlineQueryResultCa

```
attribute), 557
                                                                                                                               HRK (aiogram.enums.currency.Currency attribute), 552
HashTag (class in aiogram.utils.formatting), 665
                                                                                                                               HTML (aiogram.enums.parse_mode.ParseMode attribute),
heading (aiogram.methods.edit message live location.EditMessageEix@Location
                                                                                                                               html_text (aiogram.types.message.Message property),
                     attribute), 512
heading (aiogram.methods.send location.SendLocation
                    attribute), 457
                                                                                                                               HUF (aiogram.enums.currency.Currency attribute), 552
heading (aiogram.types.inline query result location.InlineQueryResultLocation
                     attribute), 315
heading (\it aiogram.types.input\_location\_message\_content. In {\tt protein} {\tt ent} {\tt input} {\tt location} {\tt message\_content}. In {\tt protein} {\tt ent} {
                                                                                                                                                                                                        (class
                                                                                                                                                                                                                                                     in
                     attribute), 330
                                                                                                                                                     aiogram.utils.i18n.middleware), 650
heading (aiogram.types.location.Location attribute),
                                                                                                                               icon_color(aiogram.methods.create forum topic.CreateForumTopic
                                                                                                                                                    attribute), 400
height(aiogram.methods.send_animation.SendAnimation
                                                                                                                               icon_color (aiogram.types.forum_topic.ForumTopic at-
                    attribute), 443
                                                                                                                                                     tribute), 138
height (aiogram.methods.send_video.SendVideo
                                                                                                                               icon_color(aiogram.types.forum topic created.ForumTopicCreated
                     tribute), 476
                                                                                                                                                    attribute), 138
height (aiogram.types.animation.Animation attribute),
                                                                                                                               icon_custom_emoji_id
                                                                                                                                                     (aiogram.methods.create_forum_topic.CreateForumTopic
height (aiogram.types.input_media_animation.InputMediaAnimationattribute), 400
                     attribute), 187
                                                                                                                               icon_custom_emoji_id
height(aiogram.types.input\_media\_video.InputMediaVideo
                                                                                                                                                     (aiogram.methods.edit_forum_topic.EditForumTopic
                     attribute), 191
                                                                                                                                                     attribute), 409
height(aiogram.types.input_paid_media_video.InputPaidMedia_Videotom_emoji_id
                     attribute), 193
                                                                                                                                                     (aiogram.types.forum_topic.ForumTopic
height (aiogram.types.paid_media_preview.PaidMediaPreview
                                                                                                                                                     attribute), 138
                     attribute), 264
                                                                                                                               icon_custom_emoji_id
height (aiogram.types.photo_size.PhotoSize attribute),
                                                                                                                                                    (aiogram.types.forum_topic_created.ForumTopicCreated
                                                                                                                                                     attribute), 138
height (aiogram.types.sticker.Sticker attribute), 334
                                                                                                                               icon_custom_emoji_id
height (aiogram.types.video.Video attribute), 279
                                                                                                                                                     (aiogram.types.forum_topic_edited.ForumTopicEdited
HIDDEN_USER (aiogram.enums.message_origin_type.MessageOriginTyperibute), 139
                     attribute), 558
                                                                                                                               id(aiogram.types.business connection.BusinessConnection
hide_url (aiogram.types.inline_query_result_article.InlineQueryResultArticle, 31
                     attribute), 288
                                                                                                                               id (aiogram.types.callback_query.CallbackQuery
HideGeneralForumTopic
                                                                                                                                                    tribute), 34
                    aiogram.methods.hide general forum topic),
                                                                                                                               id (aiogram.types.chat.Chat attribute), 36
                                                                                                                               id (aiogram.types.chat_full_info.ChatFullInfo attribute),
HKD (aiogram.enums.currency.Currency attribute), 552
HNL (aiogram.enums.currency.Currency attribute), 552
                                                                                                                               id (aiogram.types.inline_query.InlineQuery attribute),
horizontal_accuracy
                                                                                                                                                     284
                     (aiogram.methods.edit\_message\_live\_location. Edit Message Live\_location E
                     attribute), 512
                                                                                                                                                     attribute), 287
horizontal_accuracy
                                                                                                                               id(aiogram.types.inline_query_result_audio.InlineQueryResultAudio
                     (aiogram.methods.send_location.SendLocation
                                                                                                                                                     attribute), 289
                     attribute), 457
                                                                                                                               id(aiogram.types.inline_query_result_cached_audio.InlineQueryResultCa
horizontal_accuracy
                                                                                                                                                    attribute), 291
                     (aiogram.types.inline\_query\_result\_location.InlineQueryResultIpesatinane\_query\_result\_cached\_document.InlineQueryResultIpesatinane
                    attribute), 315
                                                                                                                                                     attribute), 294
horizontal_accuracy
                                                                                                                               id(aiogram.types.inline_query_result_cached_gif.InlineQueryResultCache
                     (aiogram.types.input_location_message_content.InputLocation_MessageGontent
                     attribute), 329
                                                                                                                               id(aiogram.types.inline_query_result_cached_mpeg4_gif.InlineQueryResu
horizontal_accuracy
                                                                                                                                                     attribute), 298
```

Index 751

attribute), 300

attribute),

(aiogram.types.location.Location

200

```
id (aiogram.types.inline_query_result_cached_sticker.InlineQueryResultsClacker)dSticker
             attribute), 302
                                                                                   inline_message_id(aiogram.methods.edit_message_media.EditMessage
id (aiogram.types.inline_query_result_cached_video.InlineQueryResult@ibahed,Vid&o
                                                                                   inline_message_id(aiogram.methods.edit_message_reply_markup.Edit]
             attribute), 304
id (aiogram.types.inline_query_result_cached_voice.InlineQueryResult@ibulue).\voice
             attribute), 305
                                                                                   inline_message_id(aiogram.methods.edit message text.EditMessageTe.
id (aiogram.types.inline query result contact.InlineQueryResultContacttibute), 517
                                                                                   inline_message_id(aiogram.methods.get_game_high_scores.GetGameF
             attribute), 307
id (aiogram.types.inline_query_result_document.InlineQueryResultDattwihente), 526
                                                                                   inline_message_id(aiogram.methods.set_game_score.SetGameScore
             attribute), 309
id(aiogram.types.inline_query_result_game.InlineQueryResultGame attribute), 529
             attribute), 311
                                                                                   inline_message_id(aiogram.methods.stop_message_live_location.Stop)
id(aiogram.types.inline_query_result_gif.InlineQueryResultGif
                                                                                                 attribute), 519
             attribute), 312
                                                                                   inline_message_id(aiogram.types.callback_query.CallbackQuery
id (aiogram.types.inline_query_result_location.InlineQueryResultLocattiohute), 35
             attribute), 314
                                                                                   inline\_message\_id ( aiogram.types.chosen\_inline\_result.ChosenInlineRe
id (aiogram.types.inline_query_result_mpeg4_gif.InlineQueryResultMapacits, 284
                                                                                   inline_message_id(aiogram.types.sent_web_app_message.SentWebApp
             attribute), 317
id(aiogram.types.inline_query_result_photo.InlineQueryResultPhotoattribute), 333
                                                                                   INLINE_QUERY (aiogram.enums.update_type.UpdateType
             attribute), 318
id (aiogram.types.inline_query_result_venue.InlineQueryResultVenueattribute), 562
             attribute), 320
                                                                                   inline_query (aiogram.types.update.Update attribute),
id(aiogram.types.inline_query_result_video.InlineQueryResultVideo361
             attribute), 323
                                                                                   inline_query_id(aiogram.methods.answer_inline_query.AnswerInlineQ
id(aiogram.types.inline_query_result_voice.InlineQueryResultVoice attribute), 522
             attribute), 324
                                                                                   InlineKeyboardBuilder
                                                                                                                                         (class
                                                                                                                                                                in
id (aiogram.types.poll.Poll attribute), 265
                                                                                                 aiogram.utils.keyboard), 644
\verb|id|| (a iogram.types.pre\_checkout\_query.PreCheckoutQuery | \verb|InlineKeyboardButton|| \\
                                                                                                                                        (class
                                                                                                                                                                in
             attribute), 339
                                                                                                 aiogram.types.inline_keyboard_button), 183
                                                                                   InlineKeyboardMarkup
id (aiogram.types.shipping_option.ShippingOption at-
                                                                                                                                        (class
                                                                                                                                                                in
              tribute), 343
                                                                                                 aiogram.types.inline_keyboard_markup),
id
       (aiogram.types.shipping_query.ShippingQuery
             tribute), 344
                                                                                   InlineQuery (class in aiogram.types.inline_query), 284
id\ (aiogram.types.star\_transaction.StarTransaction\ at-
                                                                                   InlineQueryResult
                                                                                                                                     (class
                                                                                                                                                                in
             tribute), 345
                                                                                                 aiogram.types.inline_query_result), 286
id (aiogram.types.story.Story attribute), 273
                                                                                   InlineQueryResultArticle
                                                                                                                                            (class
                                                                                                                                                                in
id (aiogram.types.user.User attribute), 275
                                                                                                 aiogram.types.inline_query_result_article),
id (aiogram.utils.web_app.WebAppChat attribute), 657
                                                                                                 287
id (aiogram.utils.web_app.WebAppUser attribute), 656
                                                                                   InlineQueryResultAudio
                                                                                                                                          (class
                                                                                                                                                                in
{\tt IDENTITY\_CARD}\ (a iogram.enums.encrypted\_passport\_element. Encry \textit{paiedPasspoypeElinhievet}\_query\_result\_audio),
             attribute), 554
IDR (aiogram.enums.currency.Currency attribute), 552
                                                                                   InlineQueryResultCachedAudio
                                                                                                                                                                in
ILS (aiogram.enums.currency.Currency attribute), 552
                                                                                                 aiogram.types.inline_query_result_cached_audio),
InaccessibleMessage
                                                    (class
                                                                             in
             aiogram.types.inaccessible_message), 143
                                                                                   InlineQueryResultCachedDocument
                                                                                                                                                  (class
include_router() (aiogram.dispatcher.router.Router
                                                                                                 aiogram.types.inline_query_result_cached_document),
                                                                                                 292
             method), 569
include_routers() (aiogram.dispatcher.router.Router InlineQueryResultCachedGif
                                                                                                                                             (class
                                                                                                                                                                 in
              method), 569
                                                                                                 aiogram.types.inline_query_result_cached_gif),
inline_keyboard(aiogram.types.inline_keyboard_markup.InlineKeyboardMarkup
             attribute), 184
                                                                                   In line Query Result Cached Mpeg 4 Gif \\
                                                                                                                                                  (class
                                                                                                                                                                in
in line\_message\_id (a iogram.methods.edit\_message\_caption. Edit \textit{Meissegne Cappian} in line\_query\_result\_cached\_mpeg4\_gif), and the line of the lin
             attribute), 510
inline_message_id (aiogram.methods.edit message live IndianeOutedit/MessukeCiveHeadPhioto
                                                                                                                                                (class
                                                                                                                                                                 in
```

aiogram.types.inline_query_result_cached_photo	o), attribute), 287 input_message_content
InlineQueryResultCachedSticker (class in	(aiogram.types.inline_query_result_audio.InlineQueryResultAudi
aiogram.types.inline_query_result_cached_sticke	er), attribute), 290
300	input_message_content
InlineQueryResultCachedVideo (class in aiogram.types.inline_query_result_cached_video	
302	input_message_content
InlineQueryResultCachedVoice (class in	(aiogram.types.inline_query_result_cached_document.InlineQuer
aiogram.types.inline_query_result_cached_voice	
304	input_message_content
<pre>InlineQueryResultContact (class in</pre>	(aiogram.types.inline_query_result_cached_gif.InlineQueryResulattribute), 296
306	input_message_content
<pre>InlineQueryResultDocument (class in</pre>	(aiogram.types.inline_query_result_cached_mpeg4_gif.InlineQueattribute), 298
308	input_message_content
<pre>InlineQueryResultGame (class in</pre>	(aiogram.types.inline_query_result_cached_photo.InlineQueryReattribute), 300
310	input_message_content
<pre>InlineQueryResultGif (class in</pre>	(aiogram.types.inline_query_result_cached_sticker.InlineQueryRattribute), 302
	input_message_content
<pre>aiogram.types.inline_query_result_location), 313</pre>	(aiogram.types.inline_query_result_cached_video.InlineQueryRe attribute), 304
InlineQueryResultMpeg4Gif (class in	input_message_content
<pre>aiogram.types.inline_query_result_mpeg4_gif), 315</pre>	(aiogram.types.inline_query_result_cached_voice.InlineQueryRe.attribute), 306
InlineQueryResultPhoto (class in	input_message_content
<pre>aiogram.types.inline_query_result_photo), 318</pre>	(aiogram.types.inline_query_result_contact.InlineQueryResultCoattribute), 308
InlineQueryResultsButton (class in	input_message_content
<pre>aiogram.types.inline_query_results_button), 325</pre>	(aiogram.types.inline_query_result_document.InlineQueryResultattribute), 310
InlineQueryResultType (class in	input_message_content
<pre>aiogram.enums.inline_query_result_type), 555</pre>	(aiogram.types.inline_query_result_gif.InlineQueryResultGif attribute), 313
InlineQueryResultVenue (class in	input_message_content
<pre>aiogram.types.inline_query_result_venue), 319</pre>	(aiogram.types.inline_query_result_location.InlineQueryResultLoattribute), 315
InlineQueryResultVideo (class in	input_message_content
aiogram.types.inline_query_result_video), 322	(aiogram.types.inline_query_result_mpeg4_gif.InlineQueryResult attribute), 317
InlineQueryResultVoice (class in	input_message_content
aiogram.types.inline_query_result_voice), 324	(aiogram.types.inline_query_result_photo.InlineQueryResultPhotoattribute), 319
input_field_placeholder	input_message_content
(aiogram.types.force_reply.ForceReply at- tribute), 137	(aiogram.types.inline_query_result_venue.InlineQueryResultVenu attribute), 321
input_field_placeholder	input_message_content
	yboardMar(aipgram.types.inline_query_result_video.InlineQueryResultVideattribute), 323
input_message_content	input_message_content
<del>_</del>	<del>_</del>

 $(a iogram. types. in line\_query\_result\_article. In lineQueryResu(\textbf{ht} Argiclen. types. in line\_query\_result\_voice. In lineQueryResult Voice and the lineQ$ 

attribute), 325	$\verb"invite_link" (a iogram.methods.edit\_chat\_subscription\_invite\_link. Edit Color of the color o$
InputContactMessageContent (class in	attribute), 408
<pre>aiogram.types.input_contact_message_content), 326</pre>	<pre>invite_link(aiogram.methods.revoke_chat_invite_link.RevokeChatInvi</pre>
InputFile (class in aiogram.types.input_file), 184	invite_link (aiogram.types.chat.Chat attribute), 38
InputInvoiceMessageContent (class in	invite_link(aiogram.types.chat_full_info.ChatFullInfo
aiogram.types.input_invoice_message_content),	attribute), 60
326	invite_link(aiogram.types.chat_invite_link.ChatInviteLink
InputLocationMessageContent (class in	attribute), 61
	invite_link(aiogram.types.chat_join_request.ChatJoinRequest
329	attribute), 63
<pre>InputMedia (class in aiogram.types.input_media), 185</pre>	$invite\_link (a iogram. types. chat\_member\_up dated. Chat Member Up dated. Chat Member $
InputMediaAnimation (class in	attribute), 110
aiogram.types.input_media_animation), 185	INVOICE (aiogram.enums.content_type.ContentType at-
InputMediaAudio (class in	tribute), 550
aiogram.types.input_media_audio), 187	invoice (aiogram.types.external_reply_info.ExternalReplyInfo
InputMediaDocument (class in	attribute), 136
aiogram.types.input_media_document), 188	invoice (aiogram.types.message.Message attribute), 208
InputMediaPhoto (class in	Invoice (class in aiogram.types.invoice), 337
aiogram.types.input_media_photo), 189	$invoice\_payload ({\it aiogram.types.pre\_checkout\_query.PreCheckoutQuery}) and {\it caiogram.types.pre\_checkout\_query.PreCheckoutQuery} and {\it caiogram.types.pre\_checkoutQuery.PreCheckoutQuery} and {\it caiogram.types.pre\_checkoutQuery.PreCheckout$
InputMediaType (class in	attribute), 339
<pre>aiogram.enums.input_media_type), 556</pre>	<pre>invoice_payload(aiogram.types.refunded_payment.RefundedPayment</pre>
InputMediaVideo (class in	attribute), 340
aiogram.types.input_media_video), 190	<pre>invoice_payload(aiogram.types.shipping_query.ShippingQuery</pre>
InputMessageContent (class in	attribute), 344
aiogram.types.input_message_content), 330	invoice_payload(aiogram.types.successful_payment.SuccessfulPayment.
InputPaidMedia (class in	attribute), 346
aiogram.types.input_paid_media), 191	<pre>invoice_payload(aiogram.types.transaction_partner_user.Transaction.</pre>
InputPaidMediaPhoto (class in	attribute), 349
aiogram.types.input_paid_media_photo),	ip_address(aiogram.methods.set_webhook.SetWebhook
192	attribute), 544
InputPaidMediaType (class in	<pre>ip_address (aiogram.types.webhook_info.WebhookInfo</pre>
aiogram.enums.input_paid_media_type),	attribute), 363
556	<pre>ip_filter_middleware()</pre>
InputPaidMediaVideo (class in	aiogram.webhook.aiohttp_server), 632
aiogram.types.input_paid_media_video),	IPFilter (class in aiogram.webhook.security), 632
192	is_animated (aiogram.types.sticker.Sticker attribute),
InputPollOption (class in	335
aiogram.types.input_poll_option), 193	is_animated (aiogram.types.sticker_set.StickerSet at-
InputSticker (class in aiogram.types.input_sticker),	tribute), 336
333	is_anonymous (aiogram.methods.promote_chat_member.PromoteChatMe
InputTextMessageContent (class in	attribute), 435
aiogram.types.input_text_message_content),	is_anonymous (aiogram.methods.send_poll.SendPoll at-
330	tribute), 471
InputVenueMessageContent (class in	is_anonymous (aiogram.types.chat_administrator_rights.ChatAdministrator
aiogram.types.input_venue_message_content),	attribute), 52
331	is_anonymous (aiogram.types.chat_member_administrator.ChatMember.
INR (aiogram.enums.currency.Currency attribute), 552	attribute), 104
	gicsupadisppallatts/taiogram.types.chat_member_owner.ChatMemberOwner
attribute), 24	attribute), 107
INTERNAL_PASSPORT (aiogram.enums.encrypted_passport	
attribute), 554	is_automatic_forward
invite_link(aiogram.methods.edit_chat_invite_link.Edi	
attribute) 406	205

```
is_big(aiogram.methods.set message reaction.SetMessageRevaideo (aiogram.types.sticker.Sticker attribute), 335
                                                        attribute), 493
                                                                                                                                                                                                                                                                                                                                    is video
                                                                                                                                                                                                                                                                                                                                                                                                                  (aiogram.types.sticker set.StickerSet
 is_blurred(aiogram.types.background type wallpaper.BackgroundTripaWullpaper
                                                        attribute), 25
                                                                                                                                                                                                                                                                                                                                    ISK (aiogram.enums.currency.Currency attribute), 552
 is_bot (aiogram.types.user.User attribute), 275
                                                                                                                                                                                                                                                                                                                                    ITALIC (aiogram.enums.message entity type.MessageEntityType
 is_bot (aiogram.utils.web app.WebAppUser attribute),
                                                                                                                                                                                                                                                                                                                                                                                           attribute), 557
                                                                                                                                                                                                                                                                                                                                    Italic (class in aiogram.utils.formatting), 666
 is_closed
                                                                                                          (aiogram.methods.send poll.SendPoll
                                                                                                                                                                                                                                                                                                                                    J
                                                        attribute), 471
 is_closed (aiogram.types.poll.Poll attribute), 265
                                                                                                                                                                                                                                                                                                                                     JMD (aiogram.enums.currency.Currency attribute), 552
 is\_disabled (\emph{a}iogram.types.link\_preview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_options.LinkPreview\_o
                                                      attribute), 199
 \verb|is_enabled| (a iogram. types. business\_connection. Business Governorm types. chat\_full\_info. ChatFullInfo and types. the property of the p
                                                      attribute), 32
                                                                                                                                                                                                                                                                                                                                                                                            attribute), 60
 is_flexible(aiogram.methods.create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_invoice_link.Create_link.Create_link.Create_link.Cr
                                                        attribute), 534
                                                                                                                                                                                                                                                                                                                                                                                           tribute), 39
 \verb|is_flexible| (a iogram.methods.send_invoice.SendInvoice | \verb|join_to_send_messages|) \\
                                                        attribute), 539
                                                                                                                                                                                                                                                                                                                                                                                           (aiogram.types.chat_full_info.ChatFullInfo
 attribute), 329
                                                                                                                                                                                                                                                                                                                                      JPY (aiogram.enums.currency.Currency attribute), 552
 is_forum (aiogram.types.chat.Chat attribute), 36
 is_forum (aiogram.types.chat full info.ChatFullInfo at-
                                                                                                                                                                                                                                                                                                                                    K
                                                        tribute), 59
                                                                                                                                                                                                                                                                                                                                    KES (aiogram.enums.currency.Currency attribute), 552
 is_from_offline (aiogram.types.message.Message at-
                                                                                                                                                                                                                                                                                                                                     keyboard (aiogram.types.reply_keyboard_markup.ReplyKeyboardMarkup
                                                        tribute), 206
                                                                                                                                                                                                                                                                                                                                                                                           attribute), 270
is_inverted(aiogram.types.background_type_pattern.BackgroundTypePattern
KeyboardButton
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               (class
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  in
                                                        attribute), 24
                                                                                                                                                                                                                                                                                                                                                                                            aiogram.types.keyboard_button), 194
 is_local (aiogram.client.telegram.TelegramAPIServer
                                                                                                                                                                                                                                                                                                                                    KeyboardButtonPollType
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          (class
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 in
                                                     attribute), 15
                                                                                                                                                                                                                                                                                                                                                                                            aiogram.types.keyboard_button_poll_type),
 is_manual
                                                                                 (aiogram.types.text_quote.TextQuote
tribute), 275

KeyboardButtonPollTypeType (class in is_member (aiogram.types.chat_member_restricted.ChatMemberRestricted_aiogram.enums.keyboard_button_poll_type_type),
                                                        tribute), 275
                                                      attribute), 108
is_moving(aiogram.types.background_type_pattern.BackgroundTypePattern
KeyboardButtonRequestChat
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 in
                                                        attribute), 24
is\_moving (a iogram.types.background\_type\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper), a iogram.types.background\_type\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper.BackgroundType\_wallpaper
                                                      attribute), 25
attribute), 25

KeyboardButtonRequestUser (class in is_persistent (aiogram.types.reply_keyboard_markup.ReplyKeyboardMarkup aiogram.types.keyboard_button_request_user),
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 in
                                                        attribute), 270
\verb|is_personal| (a iogram.methods.answer_inline\_query. AnswerInlineOuery \\ \texttt{KeyboardButtonRequestUsers}
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       (class
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 in
                                                     attribute), 522
                                                                                                                                                                                                                                                                                                                                                                                            aiogram.types.keyboard_button_request_users),
 is_premium (aiogram.types.user.User attribute), 276
                                                                                                           (aiogram.utils.web app.WebAppUser
 is premium
                                                                                                                                                                                                                                                                                                                                    KeyBuilder (class in aiogram.fsm.storage.base), 601
\verb|keywords| (a iogram. methods. set\_sticker\_keywords. SetStickerKeywords]| \\ | is\_primary| (a iogram. types. chat\_invite\_link. ChatInviteLink|) \\ | is\_primary| (a iogram. types. chat\_invite\_link|) \\ | is\_primary| (a iogram. types. chat\_invi
\verb|keywords| (a iogram.types.chat\_invite\_link.ChatInviteLink| keywords| (a iogram.types.input\_sticker.InputSticker| attacker.InputSticker| attacker| at
                                                      attribute), 62
                                                                                                                                                                                                                                                                                                                                      KGS (aiogram.enums.currency.Currency attribute), 552
\verb|is_star_giveaway| (a iogram. types. giveaway\_completed. \textit{Giveaway\_Completed}. \textit{Giveaw
                                                      attribute), 141
                                                                                                                                                                                                                                                                                                                                                                                            attribute), 548
 is_topic_message
                                                                                                                                   (aiogram.types.message.Message
                                                                                                                                                                                                                                                                                                                                    KRW (aiogram.enums.currency.Currency attribute), 552
                                                      attribute), 205
\texttt{KZT}\ (aiogram.enums.currency.Currency\ attribute), 552 \\ \texttt{is\_unclaimed}\ (aiogram.types.chat\_boost\_source\_giveaway.ChatBoostSourceGiveaway}), 552 \\ \texttt{is\_unclaimed}\ (aiogram.types.chat\_boost\_source\_giveaway.ChatBoostSourceGiveaway)}, 552 \\ \texttt{is\_unclaimed}\ (aiogram.types.chat\_boost\_source\_giveaway.ChatBoostSourceGiveaway)}, 553 \\ \texttt{is\_unclaimed}\ (aiogram.types.chat\_boost\_source\_giveaway.ChatBoostSourceGiveaway)}, 553 \\ \texttt{is\_unclaimed}\ (aiogram.types.chat\_boost\_source\_giveaway.ChatBoostSourceGiveaway)}, 554 \\ \texttt{is\_unclaimed}\ (aiogram.types.chat\_boost\_source\_giveaway.ChatBoost\_source\_giveaway)}, 555 \\ \texttt{is\_unclaimed}\ (aiogram.types.chat\_boost\_source\_giveaway.ChatBoost\_source\_giveaway.ChatBoost\_source\_giveaway)}, 555 \\ \texttt{is\_unclaimed}\ (aiogram.types.chat\_boost\_source\_giveaway.ChatBoost\_source\_giveaway)}, 555 \\ \texttt{is\_unclaimed}\ (aiogram.types.chat\_boost\_source\_giveaway.ChatBoost\_source\_giveaway)}, 555 \\ \texttt{is\_unclaimed}\ (aiogram.types.chat\_boost\_source\_giveaway.ChatBoost\_source\_giveaway)}, 555 \\ \texttt{is\_unclaimed}\ (aiogram.types.chat\_boost\_source\_giveaway)}, 555 \\ \texttt{is\_unclaimed}\ (aiogram.types.chat\_boost\_source\_giveaway)}, 555 \\ \texttt{is\_unclaimed}\ (aiogram.types.chat\_boost\_source\_giveaw
                                                      attribute), 57
```

L		attribute), 457
label	$(a iogram. types. labeled\_price. Labeled Price$	latitude (aiogram.methods.send_venue.SendVenue at-
	attribute), 337	tribute), 473
Labeled	Price (class in aiogram.types.labeled_price), 337	latitude (aiogram.types.inline_query_result_location.InlineQueryResult_attribute), 314
	ge (aiogram.types.message_entity.MessageEntity attribute), 258	latitude (aiogram.types.inline_query_result_venue.InlineQueryResultVeattribute), 320
	ge_code (aiogram.methods.delete_my_commands.	Deteitinge (giogram types.input_location_message_content.InputLocation_attribute), 329
	ge_code (aiogram.methods.get_my_commands.Ge	etMgEituda (gjogram.types.input_venue_message_content.InputVenueMess attribute), 332
	pe_code (aiogram.methods.get_my_description.Ge	ethybescription (gjogram.types.location.Location attribute), 200
languag	e_code(aiogram.methods.get my name.GetMyN	Valle (aiogram.enums.currency.Currency attribute), 552
	attribute) 427	reave() (alogram.jsm.scene.scene wizara meinoa), 020
languag	e_code (aiogram.methods.get_my_short_descript	tion eye (Ay Saio argen types chat. Chat method), 45 LeaveChat (class in aiogram.methods.leave_chat), 431
,	attribute), 428	LEFT (aigoram enums chat member status ChatMemberStatus
	atterilar to 101	MyEommanum.enums.chat_member_status.ChatMemberStatus attribute), 548
	attribute) 496	attribute), 550
languag	e_code (aiogram.methods.set_my_name.SetMyNa	ameft_chat_member (aiogram.types.message.Message attribute), 207
-	attribute), 497	
Languag	e_code (aiogram.methods.set_my_short_descript attribute), 498	ti Jeseth (singress methods send_video_note.SendVideoNote attribute), 479
languag	ge_code (aiogram.types.user.User attribute), 276	length (aiogram.types.message_entity.MessageEntity at-
languag	pe_code (aiogram.utils.web_app.WebAppUser	tribute), 257
	attribute), 656	length (aiogram.types.video_note.VideoNote attribute),
last_er	<pre>rror_date(aiogram.types.webhook_info.Webhook attribute), 363</pre>	kInfo 281 limit(aiogram.methods.get_star_transactions.GetStarTransactions
last er	ror_message(aiogram.types.webhook_info.Webb	hookInfo attribute), 535
	attribute), 363	limit (aiogram.methods.get_updates.GetUpdates at-
last_na	me (aiogram.methods.send_contact.SendContact	tribute), 542
	attribute), 450	${\tt limit} (a iogram. methods. get\_user\_profile\_photos. GetUserProfilePhotos$
last_na	me (aiogram.types.chat.Chat attribute), 36	attribute), 430
last_na	me (aiogram.types.chat_full_info.ChatFullInfo attribute), 59	link_preview(aiogram.client.default.DefaultBotProperties attribute), 567
last na	me (aiogram.types.contact.Contact attribute),	link_preview_is_disabled
1431_116	132	(aiogram.client.default.DefaultBotProperties
last na	tme (aiogram.types.inline_query_result_contact.In	aline Over Rettribute no 567
	attribute) 307	TINK_preview_options
last_na	ame (aiogram.types.input_contact_message_content attribute), 326	nt.InputContactMessageContendit_message_text.EditMessageText attribute), 517
last na	me (aiogram.types.shared_user.SharedUser at-	link_preview_options
rusc_ne	tribute), 273	(aiogram.methods.send_message.SendMessage
last_na	me (aiogram.types.user.User attribute), 275	attribute), 462
last_na		link_preview_options
	tribute), 656	(aiogram.types.external_reply_info.ExternalReplyInfo
last_sy	nchronization_error_date	attribute), 135
	(aiogram.types.webhook_info.WebhookInfo	link_preview_options
	attribute), 363	(aiogram.types.input_text_message_content.InputTextMessageC
latitud	le (aiogram.methods.edit_message_live_location.Eattribute), 512	EditMessagettVetVecuion link_preview_options
latitud	le (aiogram.methods.send_location.SendLocation	(aiogram.types.message.Message attribute),

206	longitude (aiogram.methods.send_location.SendLocation attribute), 457
link_preview_prefer_large_media (aiogram.client.default.DefaultBotProperties	longitude (aiogram.methods.send_venue.SendVenue at-
attribute), 567	tribute), 473
link_preview_prefer_small_media (aiogram.client.default.DefaultBotProperties	longitude (aiogram.types.inline_query_result_location.InlineQueryResulattribute), 314
attribute), 567	longitude (aiogram.types.inline_query_result_venue.InlineQueryResultVe
link_preview_show_above_text	attribute), 320
(aiogram.client.default.DefaultBotProperties attribute), 567	longitude (aiogram.types.input_location_message_content.InputLocation attribute), 329
linked_chat_id (aiogram.types.chat.Chat attribute),	longitude (aiogram.types.input_venue_message_content.InputVenueMess attribute), 332
linked_chat_id(aiogram.types.chat_full_info.ChatFullInfo.	
attribute), 61	200
LinkPreviewOptions (class in	
aiogram.types.link_preview_options), 199	M
${\tt live\_period} ({\it aiogram.methods.edit\_message\_live\_location}) and {\tt live\_period} ({\it aiogram.methods.edit\_me$	o <b>NAD</b> d(tNbgskagelningls:cationcy.Currency attribute), 552
attribute), 512	<pre>magic_data (aiogram.filters.magic_data.MagicData at-</pre>
${\tt live\_period} (a iogram.methods.send\_location.SendLocat$	**
attribute), 457	magic_result (aiogram.filters.command.CommandObject
live_period(aiogram.types.inline_query_result_location	
attribute), 315	MagicData (class in aiogram.filters.magic_data), 586
attribute), 330	n <b>tankhyvethaesri</b> (ii)M <b>leissgran</b> Cettient.session.base.BaseSession method), 15
	MARKDOWN (aiogram.enums.parse_mode.ParseMode at-
tribute), 200	tribute), 559
LKR (aiogram.enums.currency.Currency attribute), 552	MARKDOWN_V2 (aiogram.enums.parse_mode.ParseMode
LOCATION (aiogram.enums.content_type.ContentType at-	attribute), 559
tribute), 550	MASK (aiogram.enums.sticker_type.StickerType attribute),
LOCATION (aiogram.enums.inline_query_result_type.Inline	
attribute), 555	mask_position(aiogram.methods.set_sticker_mask_position.SetStickerMe
location (aiogram.types.business_location.BusinessLoca	
attribute), 32 location (aiogram.types.chat.Chat attribute), 39	mask_position(aiogram.types.input_sticker.InputSticker
location (aiogram.types.chat_full_info.ChatFullInfo at-	attribute), 333 mask_position(aiogram.types.sticker.Sticker attribute),
tribute), 61	335
location (aiogram.types.chat_location.ChatLocation attribute), 103	334
location (aiogram.types.chosen_inline_result.ChosenInli attribute), 284	
	aiogram.enums.mask_position_point), 557  linktgo_connections (aiogram.methods.set_webhook.SetWebhook
attribute), 136	attribute), 544
location (aiogram.types.inline_query.InlineQuery at- tribute), 285	max_connections (aiogram.types.webhook_info.WebhookInfo attribute), 363
location (aiogram.types.message.Message attribute),	max_quantity (aiogram.types.keyboard_button_request_users.KeyboardB
207	attribute), 198
location (aiogram.types.venue.Venue attribute), 279	max_reaction_count(aiogram.types.chat_full_info.ChatFullInfo
Location (class in aiogram.types.location), 200	attribute), 59
login_url(aiogram.types.inline_keyboard_button.Inlinelattribute), 183	KandxoardButtaount (aiogram.methods.create_invoice_link.CreateInvoiceLine attribute), 533
LoginUrl (class in aiogram.types.login_url), 200	max_tip_amount(aiogram.methods.send_invoice.SendInvoice
LogOut (class in aiogram.methods.log_out), 432	attribute), 538
	Faciti Messpesal content in the invoice message content. Input Invoice message content. Input Inv
attribute), 512	attribute), 328

```
MaybeInaccessibleMessage
                                                                                                                                                            (class
                                                                                                                                                                                                                                                                             attribute), 487
                                      aiogram.types.maybe_inaccessible_message),
                                                                                                                                                                                                                                      MenuButton (class in aiogram.types.menu_button), 201
                                                                                                                                                                                                                                      MenuButtonCommands
md_text (aiogram.types.message.Message property),
                                                                                                                                                                                                                                                                              aiogram.types.menu_button_commands),
                                                                                                                                                                                                                                                                              202
MDL (aiogram.enums.currency.Currency attribute), 552
                                                                                                                                                                                                                                      MenuButtonDefault
                                                                                                                                                                                                                                                                                                                                                                                (class
                                                                                                                                                                                                                                                                                                                                                                                                                                                            in
media(aiogram.methods.edit message media.EditMessageMedia
                                                                                                                                                                                                                                                                             aiogram.types.menu button default), 202
                                      attribute), 514
                                                                                                                                                                                                                                      MenuButtonType
                                                                                                                                                                                                                                                                                                                                                                          (class
                                                                                                                                                                                                                                                                                                                                                                                                                                                            in
media(aiogram.methods.send_media_group.SendMediaGroup
                                                                                                                                                                                                                                                                             aiogram.enums.menu_button_type), 557
                                                                                                                                                                                                                                       MenuButtonWebApp
                                      attribute), 460
                                                                                                                                                                                                                                                                                                                                                                              (class
                                                                                                                                                                                                                                                                                                                                                                                                                                                            in
media (aiogram.methods.send_paid_media.SendPaidMedia
                                                                                                                                                                                                                                                                             aiogram.types.menu_button_web_app), 203
                                       attribute), 464
                                                                                                                                                                                                                                      MESSAGE (aiogram.enums.update_type.UpdateType at-
media (aiogram.types.input_media_animation.InputMediaAnimation tribute), 562
                                     attribute), 186
                                                                                                                                                                                                                                      message (aiogram.handlers.callback_query.CallbackQueryHandler
media (aiogram.types.input_media_audio.InputMediaAudio
                                                                                                                                                                                                                                                                            property), 639
                                      attribute), 187
                                                                                                                                                                                                                                       message (aiogram.types.business_intro.BusinessIntro at-
media (aiogram.types.input_media_document.InputMediaDocument_tribute), 32
                                      attribute), 188
                                                                                                                                                                                                                                      message (aiogram.types.callback_query.CallbackQuery
media(aiogram.types.input_media_photo.InputMediaPhoto
                                                                                                                                                                                                                                                                             attribute), 34
                                      attribute), 189
                                                                                                                                                                                                                                       message(aiogram.types.passport_element_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportEl
media(aiogram.types.input_media_video.InputMediaVideo
                                                                                                                                                                                                                                                                             attribute), 353
                                      attribute), 190
                                                                                                                                                                                                                                      message (aiogram.types.passport_element_error_file.PassportElementErro
media (aiogram.types.input_paid_media_photo.InputPaidMediaPhotattribute), 354
                                      attribute), 192
                                                                                                                                                                                                                                      message (aiogram.types.passport_element_error_files.PassportElementErr
media (aiogram.types.input_paid_media_video.InputPaidMediaVideoattribute), 354
                                      attribute), 192
                                                                                                                                                                                                                                      message (aiogram.types.passport_element_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportE
media_group_id (aiogram.types.message.Message at-
                                                                                                                                                                                                                                                                              attribute), 355
                                     tribute), 206
                                                                                                                                                                                                                                      message (aiogram.types.passport_element_error_reverse_side.PassportEle
MediaGroupBuilder
                                                                                                                                                                                                                     in
                                                                                                                                                                                                                                                                             attribute), 356
                                                                                                                                          (class
                                      aiogram.utils.media_group), 669
                                                                                                                                                                                                                                      message (a iogram.types.passport\_element\_error\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.PassportElementError\_selfie.Pa
MEMBER (aiogram.enums.chat_member_status.ChatMemberStatus
                                                                                                                                                                                                                                                                             attribute), 357
                                      attribute), 548
                                                                                                                                                                                                                                      message (aiogram.types.passport_element_error_translation_file.Passport.
member_limit(aiogram.methods.create_chat_invite_link.CreateChatItvibalten)k358
                                                                                                                                                                                                                                      {\tt message}\ (aiogram.types.passport\_element\_error\_translation\_files.Passport\_element\_error\_translation\_files.Passport\_element\_error\_translation\_files.Passport\_element\_error\_translation\_files.Passport\_element\_error\_translation\_files.Passport\_element\_error\_translation\_files.Passport\_element\_error\_translation\_files.Passport\_element\_error\_translation\_files.Passport\_element\_error\_translation\_files.Passport\_element\_error\_translation\_files.Passport\_element\_error\_translation\_files.Passport\_element\_error\_translation\_files.Passport\_element\_error\_translation\_files.Passport\_element\_error\_translation\_files.Passport\_element\_error\_translation\_files.Passport\_element\_error\_translation\_files.Passport\_element\_error\_translation\_files.Passport\_element\_error\_translation\_files.Passport\_element\_error\_translation\_files.Passport\_element\_error\_translation\_files.Passport\_element\_error\_translation\_files.Passport\_element\_error\_translation\_files.Passport\_element\_error\_translation\_files.Passport\_element\_error\_translation\_files.Passport\_element\_error\_translation\_files.Passport\_element\_error\_translation\_files.Passport\_element\_error\_translation\_files.Passport\_element\_error\_translation\_files.Passport\_element\_error\_translation\_element\_error\_translation\_element\_error\_translation\_element\_error\_translation\_element\_error\_translation\_element\_error\_translation\_element\_error\_translation\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element\_element
                                      attribute), 396
member_limit(aiogram.methods.edit_chat_invite_link.EditChatInviteatinibute), 359
                                     attribute), 407
                                                                                                                                                                                                                                      message (aiogram.types.passport_element_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_
member_limit(aiogram.types.chat_invite_link.ChatInviteLink
                                                                                                                                                                                                                                                                             attribute), 359
                                      attribute), 62
                                                                                                                                                                                                                                      message (aiogram.types.update.Update attribute), 361
member_status_changed
                                                                                                                                                                                                                                      Message (class in aiogram.types.message), 203
                                      (aiogram.filters.chat\_member\_updated.ChatMemb{\it mers}{\it spalgee}{\it chatMor}{\it time}(aiogram.types.chat.ChatMemb{\it mers}{\it time}(aiogram.types.chatMemb{\it mers}{\it time}(aiogram.type
                                                                                                                                                                                                                                                                             attribute), 39
                                      attribute), 581
MemoryStorage (class in aiogram.fsm.storage.memory), message_auto_delete_time
                                                                                                                                                                                                                                                                             (aiogram.types.chat_full_info.ChatFullInfo
MENTION (aiogram.enums.message_entity_type.MessageEntityType
                                                                                                                                                                                                                                                                            attribute), 60
                                      attribute), 557
                                                                                                                                                                                                                                      message_auto_delete_time
mention (aiogram.filters.command.CommandObject at-
                                                                                                                                                                                                                                                                             (aiogram.types.message_auto_delete_timer_changed.MessageAut
                                      tribute), 580
                                                                                                                                                                                                                                                                             attribute), 257
                                                                                                                                                                                                                                      MESSAGE_AUTO_DELETE_TIMER_CHANGED
mention_html() (aiogram.types.user.User method),
                                       276
                                                                                                                                                                                                                                                                              (aiogram.enums.content_type.ContentType
mention_markdown()
                                                                                                                           (aiogram.types.user.User
                                                                                                                                                                                                                                                                             attribute), 550
                                     method), 276
                                                                                                                                                                                                                                      message_auto_delete_timer_changed
mentioned (aiogram.filters.command.CommandObject
                                                                                                                                                                                                                                                                             (aiogram.types.message.Message
                                                                                                                                                                                                                                                                                                                                                                                                                             attribute),
                                     property), 580
menu_button(aiogram.methods.set chat menu button.Set@bsstdgeuBfiffeat_id(aiogram.methods.send animation.SendAnimation
```

```
attribute), 443
                                                                attribute), 529
message_effect_id(aiogram.methods.send_audio.SendAmdissage_id(aiogram.methods.set_message_reaction.SetMessageReaction
                                                                attribute), 492
        attribute), 446
message_effect_id(aiogram.methods.send_contact.Sendtoessage_id(aiogram.methods.stop_message_live_location.StopMessageL
         attribute), 450
                                                                attribute), 519
message_effect_id(aiogram.methods.send dice.SendDiamessage_id
                                                                         (aiogram.methods.stop\_poll.StopPoll
         attribute), 452
                                                                attribute), 520
message_effect_id(aiogram.methods.send_document.SendDsageucind(aiogram.methods.unpin_chat_message.UnpinChatMessage
         attribute), 455
                                                                attribute), 506
message_effect_id(aiogram.methods.send_game.SendGmessage_id(aiogram.types.external_reply_info.ExternalReplyInfo
        attribute), 527
                                                                attribute), 135
message_effect_id(aiogram.methods.send_invoice.Send\text{hessizege_id}(aiogram.types.inaccessible_message.InaccessibleMessage
        attribute), 539
                                                                attribute), 143
message_effect_id (aiogram.methods.send_location.Senathessation_id (aiogram.types.message.Message attribute),
         attribute), 458
                                                                205
message_effect_id(aiogram.methods.send_media_groupm&sstdfediaGhoupgram.types.message_id.MessageId at-
        attribute), 460
                                                                tribute), 258
message_effect_id(aiogram.methods.send_message.Senmlexessage_id(aiogram.types.message_origin_channel.MessageOriginChan
        attribute), 463
                                                                attribute), 259
message_effect_id(aiogram.methods.send_photo.SendPhotssage_id(aiogram.types.message_reaction_count_updated.MessageRea
        attribute), 468
                                                                attribute), 261
message_effect_id(aiogram.methods.send_poll.SendPolmessage_id(aiogram.types.message_reaction_updated.MessageReactionU
                                                                attribute), 262
         attribute), 471
message_effect_id(aiogram.methods.send_sticker.Send\sticker.Send\sticker.ge_id(aiogram.types.reply_parameters.ReplyParameters
         attribute), 374
                                                                attribute), 271
message_effect_id(aiogram.methods.send_venue.SendVmessage_ids(aiogram.methods.copy_messages.CopyMessages
         attribute), 474
                                                                attribute), 395
message_effect_id(aiogram.methods.send_video.SendVinlessage_ids(aiogram.methods.delete_messages.DeleteMessages
        attribute), 477
                                                                attribute), 508
message_effect_id(aiogram.methods.send_video_note.SmedVidpoNicdes(aiogram.methods.forward_messages.ForwardMessages
         attribute), 480
                                                                attribute), 414
message_effect_id(aiogram.methods.send_voice.SendVoicessage_ids(aiogram.types.business_messages_deleted.BusinessMessage
        attribute), 482
                                                                attribute), 33
message_id(aiogram.methods.copy_message.CopyMessagMESSAGE_REACTION(aiogram.enums.update_type.UpdateType
        attribute), 392
                                                                attribute), 562
message_id(aiogram.methods.delete_message.DeleteMessmessage_reaction (aiogram.types.update.Update at-
        attribute), 507
                                                                tribute), 361
message_id (aiogram.methods.edit_message_caption.EditMESSAGE_aRFAGTION_COUNT
         attribute), 510
                                                                (aiogram.enums.update_type.UpdateType
message_id (aiogram.methods.edit_message_live_location.EditMessage_liveL)ocation
        attribute), 512
                                                      message_reaction_count
message_id(aiogram.methods.edit_message_media.EditMessageMediaiogram.types.update.Update
                                                                                                  attribute),
        attribute), 514
message_id(aiogram.methods.edit_message_reply_markumEslitAlgessugrReplyMourkutppes.input_text_message_content.InputTextMessa
         attribute), 515
                                                                attribute), 331
message_id(aiogram.methods.edit_message_text.EditMessage_thread_id(aiogram.methods.close_forum_topic.CloseForumTo
        attribute), 517
                                                                attribute), 390
message_id(aiogram.methods.forward_message.ForwardMessage_thread_id(aiogram.methods.copy_message.CopyMessage
        attribute), 412
                                                                attribute), 393
message_id(aiogram.methods.get_game_high_scores.Get@nessMgeh&hread_id(aiogram.methods.copy_messages.CopyMessages
        attribute), 526
                                                                attribute), 395
```

attribute), 433

message\_id(aiogram.methods.pin chat message.PinChatMessage\_thread\_id(aiogram.methods.delete forum topic.DeleteForumT

message\_id(aiogram.methods.set game score.SetGameScmessage\_thread\_id(aiogram.methods.edit forum topic.EditForumTopic

attribute), 404

	ssageEntityType	(class	in
message_thread_id(aiogram.methods.forward_message.Foattribute), 413	rwaraM <b>æxsgræ</b> m.enums.message <sub>-</sub> ssageId ( <i>class in aiogram.type</i> .		
message_thread_id(aiogram.methods.forward_messagesMe		class	in
attribute), 414	aiogram.types.message_c		
message_thread_id(aiogram.methods.reopen_forum_topNe		(class	in
attribute), 437	aiogram.types.message_o	`	in
${\tt message\_thread\_id} \ (a iogram. methods. send\_animation. $	lAnimat <b>īlōn</b>		
attribute), 442	ssageOriginChat	(class	in
<pre>message_thread_id(aiogram.methods.send_audio.SendAudionale.</pre>	io aiogram.types.message_o	origin_chat), 259	
attribute), 445	ssageOriginHiddenUser	(class	in
message_thread_id(aiogram.methods.send_chat_action.Send_c		origin hidden user`	).
attribute), 448	260	0 = = /	•
message_thread_id(aiogram.methods.send_contact.Send)Ce		(class	in
attribute), 450	aiogram.enums.message_	*	iii
message_thread_id(aiogram.methods.send_dice.SendDidee		_origin_type), 556 (class	in
		`	in
attribute), 452	aiogram.types.message_c		
message_thread_id(aiogram.methods.send_document.SeMa	_		in
attribute), 454	aiogram.types.message_1	reaction_count_upd	ated),
${\tt message\_thread\_id} \ (a iogram. methods. send\_game. Send Games and Send Game$			
attribute), 527	ssageReactionUpdated	(class	in
<pre>message_thread_id(aiogram.methods.send_invoice.SendInv</pre>	voice aiogram.types.message_n	reaction_updated),	
attribute), 538	262		
message_thread_id(aiogram.methods.send_location.SenML	GRATIEN FROM CHAT ID		
attribute), 457	(aiogram.enums.content_	type ContentType	
message_thread_id(aiogram.methods.send_media_group.S	. •	iype.comemiype	
	grate_from_chat_id		
message_thread_id(aiogram.methods.send_message.SendM		Message attribu	(a)
		message anribu	<i>(e)</i> ,
attribute), 462	208		C , T
message_thread_id(aiogram.methods.send_photo.SendPMa		nums.content_type.	ContentType
attribute), 467	attribute), 550		
${\tt message\_thread\_id} (a iogram.methods.send\_poll.SendPoll{\tt missage\_thread\_id})$		ypes.message.Messa	ige
attribute), 470	attribute), 208		
message_thread_id(aiogram.methods.send_sticker.SendSmix	<pre>grate_to_chat_id(aiogram.t</pre>	ypes.response_para	meters.ResponsePara
attribute), 373	attribute), 272		
message_thread_id(aiogram.methods.send_venue.SendVmi	me_type (aiogram.types.anin	nation.Animation	at-
attribute), 473	tribute), 20		
message_thread_id(aiogram.methods.send_video.SendVimb		Audio attribute). 20	
- "	me_type (alogram.types.doci		at-
message_thread_id(aiogram.methods.send_video_note.Sen		imeni.Documeni	ш
		arram, magnit da arra	and Inlin of warm. Dog
	me_type(aiogram.types.inline_	query_resuit_aocun	ient.IntineQuerykest
message_thread_id(aiogram.methods.send_voice.SendVoic	**		
	me_type(aiogram.types.inline_		InlineQueryResultVi
${\tt message\_thread\_id} ({\it aiogram.methods.unpin\_all\_forum\_top})$			
attribute), 504 mi	<pre>me_type (aiogram.types.video.V</pre>	<sup>7</sup> ideo attribute), <mark>28</mark> 0	
<pre>message_thread_id(aiogram.types.forum_topic.ForumTopia</pre>	${\tt me\_type}~(aiogram.types.voice.V)$	oice attribute), 282	
attribute), 137	T (aiogram.enums.currency.Cur.	rency attribute), 553	3
<pre>message_thread_id (aiogram.types.message.Message mo</pre>	del_computed_fields	•	
attribute), 205	(aiogram.filters.callback	data.CallbackData	
MessageAutoDeleteTimerChanged (class in	attribute), 587		
aiogram.types.message_auto_delete_timer_changead	**		
257	(aiogram.methods.add_s	ticker to set AddSti	ckerToSet
MessageEntity (class in	attribute), 365	ccrio_sei1uusii	CIMI TUDEI
aiogram.types.message_entity), 257 mo	del_computed_fields		

```
(aiogram.methods.answer_callback_query.AnswerCallbackQniegram.methods.decline_chat_join_request.DeclineChatJoinRe
              attribute), 384
                                                                                                       attribute), 401
model_computed_fields
                                                                                        model_computed_fields
               (aiogram.methods.answer_inline_query.AnswerInlineQuery(aiogram.methods.delete_chat_photo.DeleteChatPhoto
              attribute), 522
                                                                                                       attribute), 402
model_computed_fields
                                                                                        model_computed_fields
              (aiogram.methods.answer pre checkout query.AnswerPreChaeokoanOmmethods.delete chat sticker set.DeleteChatStickerSet
              attribute), 530
                                                                                                       attribute), 403
model_computed_fields
                                                                                        model_computed_fields
              (aiogram.methods.answer_shipping_query.AnswerShippingQuiogram.methods.delete_forum_topic.DeleteForumTopic
              attribute), 531
                                                                                                       attribute), 404
model_computed_fields
                                                                                        model_computed_fields
               (aiogram.methods.answer\_web\_app\_query. AnswerWebApp \textbf{Quinerg} ram.methods. delete\_message. DeleteMessage
              attribute), 524
                                                                                                       attribute), 507
model_computed_fields
                                                                                        model_computed_fields
              (aiogram.methods.approve_chat_join_request.ApproveChat.lainReamestethods.delete_messages.DeleteMessages
              attribute), 385
                                                                                                       attribute), 508
model_computed_fields
                                                                                        model_computed_fields
              (aiogram.methods.ban chat member.BanChatMember
                                                                                                       (aiogram.methods.delete_my_commands.DeleteMyCommands
              attribute), 386
                                                                                                       attribute), 405
model_computed_fields
                                                                                        model_computed_fields
              (aiogram.methods.ban_chat_sender_chat.BanChatSenderCkatogram.methods.delete_sticker_from_set.DeleteStickerFromSet
              attribute), 388
                                                                                                        attribute), 368
model_computed_fields
                                                                                        model_computed_fields
                                                                                                       (aiogram.methods.delete\_sticker\_set.DeleteStickerSet
              (aiogram.methods.close.Close
                                                                      attribute),
               389
                                                                                                       attribute), 369
model_computed_fields
                                                                                        model_computed_fields
               (aiogram.methods.close_forum_topic.CloseForumTopic
                                                                                                        (aiogram.methods.delete_webhook.DeleteWebhook
              attribute), 390
                                                                                                       attribute), 541
model_computed_fields
                                                                                        model_computed_fields
              (aiogram.methods.close_general_forum_topic.CloseGeneral_foognaffoomiethods.edit_chat_invite_link.EditChatInviteLink
              attribute), 391
                                                                                                        attribute), 406
model_computed_fields
                                                                                        model_computed_fields
               (aiogram.methods.copy_message.CopyMessage
                                                                                                       (aiogram.methods.edit\_chat\_subscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubscription\_invite\_link.EditChatSubs
              attribute), 393
                                                                                                       attribute), 408
model_computed_fields
                                                                                        model_computed_fields
              (aiogram.methods.copy messages.CopyMessages
                                                                                                       (aiogram.methods.edit forum topic.EditForumTopic
              attribute), 395
                                                                                                       attribute), 409
model_computed_fields
                                                                                        model_computed_fields
              (aiogram.methods.create_chat_invite_link.CreateChatInvitelaidgram.methods.edit_general_forum_topic.EditGeneralForumT
              attribute), 396
                                                                                                       attribute), 410
model_computed_fields
                                                                                        model_computed_fields
               (aiogram.methods.create chat subscription invite link.CretateGraffuthethintsconditivitedsiande caption.EditMessageCaption
              attribute), 398
                                                                                                       attribute), 510
model_computed_fields
                                                                                        model_computed_fields
              (aiogram.methods.create_forum_topic.CreateForumTopic (aiogram.methods.edit_message_live_location.EditMessageLiveL
              attribute), 399
                                                                                                       attribute), 512
model_computed_fields
                                                                                        model_computed_fields
              (aiogram.methods.create_invoice_link.CreateInvoiceLink
                                                                                                       (aiogram.methods.edit_message_media.EditMessageMedia
              attribute), 534
                                                                                                       attribute), 514
model_computed_fields
                                                                                        model_computed_fields
              (aiogram.methods.create_new_sticker_set.CreateNewSticker(Suiogram.methods.edit_message_reply_markup.EditMessageRepl
              attribute), 367
                                                                                                       attribute), 515
model_computed_fields
                                                                                        model_computed_fields
```

```
(aiogram.methods.get my name.GetMyName
        (aiogram.methods.edit message text.EditMessageText
        attribute), 517
                                                              attribute), 427
model_computed_fields
                                                     model_computed_fields
         (aiogram.methods.export_chat_invite_link.ExportChatInvite/laiiolgram.methods.get_my_short_description.GetMyShortDescrip
         attribute), 411
                                                              attribute), 428
model_computed_fields
                                                     model_computed_fields
        (aiogram.methods.forward message.ForwardMessage
                                                              (aiogram.methods.get star transactions.GetStarTransactions
        attribute), 413
                                                              attribute), 535
model_computed_fields
                                                     model_computed_fields
        (aiogram.methods.forward_messages.ForwardMessages
                                                              (aiogram.methods.get_sticker_set.GetStickerSet
        attribute), 414
                                                              attribute), 371
model_computed_fields
                                                     model_computed_fields
         (aiogram.methods.get_business_connection.GetBusinessCon(neoptian.methods.get_updates.GetUpdates
        attribute), 415
                                                              attribute), 542
model_computed_fields
                                                     model_computed_fields
        (aiogram.methods.get_chat.GetChat attribute),
                                                              (aiogram.methods.get_user_chat_boosts.GetUserChatBoosts
        416
                                                              attribute), 429
                                                     model_computed_fields
model_computed_fields
        (aiogram.methods.get_chat_administrators.GetChatAdministrations.methods.get_user_profile_photos.GetUserProfilePhotos
        attribute), 417
                                                              attribute), 429
model_computed_fields
                                                     model_computed_fields
        (aiogram.methods.get chat member.GetChatMember
                                                              (aiogram.methods.get_webhook_info.GetWebhookInfo
        attribute), 418
                                                              attribute), 543
model_computed_fields
                                                     model_computed_fields
        (aiogram.methods.get_chat_member_count.GetChatMembetGingram.methods.hide_general_forum_topic.HideGeneralForum
        attribute), 419
                                                              attribute), 430
model_computed_fields
                                                     model_computed_fields
         (aiogram.methods.get_chat_menu_button.GetChatMenuButtanogram.methods.leave_chat.LeaveChat
                                                              attribute), 431
        attribute), 420
model_computed_fields
                                                     model_computed_fields
        (aiogram.methods.get_custom_emoji_stickers.GetCustomEmajiSgircharmethods.log_out.LogOut attribute),
        attribute), 370
                                                              432
model_computed_fields
                                                     model_computed_fields
        (aiogram.methods.get_file.GetFile
                                                              (aiogram.methods.pin_chat_message.PinChatMessage
                                          attribute),
                                                              attribute), 433
         421
model_computed_fields
                                                     model_computed_fields
        (aiogram.methods.get forum topic icon stickers.GetForum(kaipiglaxonus)titikeds.promote chat member.PromoteChatMember
        attribute), 422
                                                              attribute), 435
model_computed_fields
                                                     model_computed_fields
        (aiogram.methods.get_game_high_scores.GetGameHighScotwiogram.methods.refund_star_payment.RefundStarPayment
        attribute), 526
                                                              attribute), 536
model_computed_fields
                                                     model_computed_fields
         (aiogram.methods.get me.GetMe
                                                              (aiogram.methods.reopen forum topic.ReopenForumTopic
                                          attribute),
         423
                                                              attribute), 437
model_computed_fields
                                                     model_computed_fields
        (aiogram.methods.get_my_commands.GetMyCommands
                                                              (aiogram.methods.reopen_general_forum_topic.ReopenGeneralF
        attribute), 424
                                                              attribute), 438
model_computed_fields
                                                     model_computed_fields
        (aiogram.methods.get_my_default_administrator_rights.GetMidQefaultMadhcidisteplacRighticker_in_set.ReplaceStickerInSet
        attribute), 425
                                                              attribute), 372
model_computed_fields
                                                     model_computed_fields
        (aiogram.methods.get my description.GetMyDescription
                                                              (aiogram.methods.restrict_chat_member.RestrictChatMember
        attribute), 426
                                                              attribute), 439
model_computed_fields
                                                     model_computed_fields
```

(aiogram.methods.revoke_chat_invite_link.Revok attribute), 441	keChatInvit <b>(hink</b> ram.methods.send_video_note.SendVideoNote attribute), 480
model_computed_fields	model_computed_fields
(aiogram.methods.send_animation.SendAnimatio	
attribute), 443	attribute), 482
model_computed_fields	model_computed_fields
(aiogram.methods.send_audio.SendAudio	(aiogram.methods.set_chat_administrator_custom_title.SetChatA
attribute), 446	attribute), 484
model_computed_fields	model_computed_fields
(aiogram.methods.send_chat_action.SendChatAc	
attribute), 448	attribute), 485
model_computed_fields	model_computed_fields
(aiogram.methods.send_contact.SendContact	(aiogram.methods.set_chat_menu_button.SetChatMenuButton
attribute), 450	attribute), 486
model_computed_fields	model_computed_fields
(aiogram.methods.send_dice.SendDice at-	(aiogram.methods.set_chat_permissions.SetChatPermissions
<i>tribute</i> ), 452	attribute), 488
model_computed_fields	model_computed_fields
(aiogram.methods.send_document.SendDocumer	
attribute), 455	attribute), 489
model_computed_fields	model_computed_fields
(aiogram.methods.send_game.SendGame	(aiogram.methods.set_chat_sticker_set.SetChatStickerSet
attribute), 527	attribute), 490
model_computed_fields	model_computed_fields
(aiogram.methods.send_invoice.SendInvoice	(aiogram.methods.set_chat_title.SetChatTitle
attribute), 538	attribute), 491
model_computed_fields	model_computed_fields
(aiogram.methods.send_location.SendLocation	(aiogram.methods.set_custom_emoji_sticker_set_thumbnail.SetC
attribute), 457	attribute), 375
model_computed_fields	model_computed_fields
(aiogram.methods.send_media_group.SendMedia	
attribute), 460	attribute), 529
model_computed_fields	model_computed_fields
(aiogram.methods.send_message.SendMessage	(aiogram.methods.set_message_reaction.SetMessageReaction
attribute), 462	attribute), 492
model_computed_fields	model_computed_fields
(aiogram.methods.send_paid_media.SendPaidMo	
attribute), 465	attribute), 494
model_computed_fields	model_computed_fields
(aiogram.methods.send_photo.SendPhoto	(aiogram.methods.set_my_default_administrator_rights.SetMyD
attribute), 467	attribute), 495
model_computed_fields	model_computed_fields
(aiogram.methods.send_poll.SendPoll at-	(aiogram.methods.set_my_description.SetMyDescription
tribute), 471	attribute), 496
model_computed_fields	model_computed_fields
(aiogram.methods.send_sticker.SendSticker	(aiogram.methods.set_my_name.SetMyName
attribute), 374	attribute), 497
model_computed_fields	model_computed_fields
(aiogram.methods.send_venue.SendVenue	(aiogram.methods.set_my_short_description.SetMyShortDescrip
attribute), 473	attribute), 498
model_computed_fields	model_computed_fields
(aiogram.methods.send_video.SendVideo	(aiogram.methods.set_passport_data_errors.SetPassportDataEr
attribute), 477	attribute), 546
model computed fields	model computed fields

```
(aiogram.methods.set_sticker_emoji_list.SetStickerEmojiListattribute), 20
        attribute), 376
                                                      model_computed_fields
model_computed_fields
                                                               (aiogram.types.background_fill.BackgroundFill
         (aiogram.methods.set_sticker_keywords.SetStickerKeywordsattribute), 21
         attribute), 377
                                                      model_computed_fields
model_computed_fields
                                                               (aiogram.types.background_fill_freeform_gradient.BackgroundFi
         (aiogram.methods.set sticker mask position.SetStickerMaskaPoribuioe), 21
                                                      model_computed_fields
         attribute), 378
model_computed_fields
                                                               (aiogram.types.background_fill_gradient.BackgroundFillGradien
         (aiogram.methods.set_sticker_position_in_set.SetStickerPosittonibuse), 22
         attribute), 379
                                                      model_computed_fields
                                                               (aiogram.types.background_fill_solid.BackgroundFillSolid
model_computed_fields
         (aiogram.methods.set_sticker_set_thumbnail.SetStickerSetTlattnlbutel), 22
        attribute), 380
                                                      model_computed_fields
model_computed_fields
                                                               (aiogram.types.background_type.BackgroundType
         (aiogram.methods.set\_sticker\_set\_title.SetStickerSetTitle
                                                               attribute), 23
        attribute), 382
                                                      model_computed_fields
model_computed_fields
                                                               (aiogram.types.background_type_chat_theme.BackgroundTypeCh
         (aiogram.methods.set_webhook.SetWebhook
                                                               attribute), 23
                                                      model_computed_fields
         attribute), 544
model_computed_fields
                                                               (aiogram.types.background_type_fill.BackgroundTypeFill
         (aiogram.methods.stop_message_live_location.StopMessageditivelbuteation)
         attribute), 519
                                                      model_computed_fields
model_computed_fields
                                                               (aiogram.types.background type pattern.BackgroundTypePattern
         (aiogram.methods.stop_poll.StopPoll attribute),
                                                               attribute), 24
                                                      model_computed_fields
model_computed_fields
                                                               (aiogram.types.background_type_wallpaper.BackgroundTypeWall
         (aiogram.methods.unban_chat_member.UnbanChatMemberattribute), 25
                                                      model_computed_fields
        attribute), 499
model_computed_fields
                                                               (aiogram.types.birthdate.Birthdate attribute),
         (aiogram.methods.unban_chat_sender_chat.UnbanChatSend&rChat
        attribute), 500
                                                      model_computed_fields
model_computed_fields
                                                               (aiogram.types.bot_command.BotCommand
         (aiogram.methods.unhide_general_forum_topic.UnhideGeneratalibata); Zopic
         attribute), 501
                                                      model_computed_fields
model_computed_fields
                                                               (aiogram.types.bot_command_scope.BotCommandScope
         (aiogram.methods.unpin_all_chat_messages.UnpinAllChatMessibnes), 26
         attribute), 502
                                                      model_computed_fields
model_computed_fields
                                                               (aiogram.types.bot_command_scope_all_chat_administrators.Bo
         (aiogram.methods.unpin_all_forum_topic_messages.UnpinAdlfributm];@plcMessages
         attribute), 503
                                                      model_computed_fields
model_computed_fields
                                                               (aiogram.types.bot_command_scope_all_group_chats.BotComma
         (aiogram.methods.unpin_all_general_forum_topic_messagesttriipninteAllCeneralForumTopicMessages
         attribute), 505
                                                      model_computed_fields
model_computed_fields
                                                               (aiogram.types.bot_command_scope_all_private_chats.BotComm
         (aiogram.methods.unpin_chat_message.UnpinChatMessageattribute), 28
                                                      model_computed_fields
        attribute), 506
model_computed_fields
                                                               (aiogram.types.bot_command_scope_chat.BotCommandScopeCh
         (aiogram.methods.upload_sticker_file.UploadStickerFile
                                                               attribute), 28
                                                      model_computed_fields
        attribute), 383
model_computed_fields
                                                               (aiogram.types.bot_command_scope_chat_administrators.BotCon
                                                               attribute), 29
         (aiogram.types.animation.Animation attribute),
                                                      model_computed_fields
model_computed_fields (aiogram.types.audio.Audio
                                                               (aiogram.types.bot command scope chat member.BotCommand
```

attribute), 30	model_computed_fields
model_computed_fields	(aiogram.types.chat_boost_source.ChatBoostSource
(aiogram.types.bot_command_scope_default.Bo	* **
attribute), 30	model_computed_fields
model_computed_fields	(aiogram.types.chat_boost_source_gift_code.ChatBoostSourceGi
(aiogram.types.bot_description.BotDescription	attribute), 56
attribute), 30	model_computed_fields
model_computed_fields	(aiogram.types.chat_boost_source_giveaway.ChatBoostSourceGi
(aiogram.types.bot_name.BotName attribute),	
31	model_computed_fields
model_computed_fields	(aiogram.types.chat_boost_source_premium.ChatBoostSourcePre
(aiogram.types.bot_short_description.BotShort)	
attribute), 31	model_computed_fields
model_computed_fields	(aiogram.types.chat_boost_updated.ChatBoostUpdated
(aiogram.types.business_connection.BusinessCo	
attribute), 31	model_computed_fields
model_computed_fields	(aiogram.types.chat_full_info.ChatFullInfo
(aiogram.types.business_intro.BusinessIntro	attribute), 60
attribute), 32	model_computed_fields
model_computed_fields	(aiogram.types.chat_invite_link.ChatInviteLink
(aiogram.types.business_location.BusinessLoca	
attribute), 32	model_computed_fields
model_computed_fields	(aiogram.types.chat_join_request.ChatJoinRequest
(aiogram.types.business_messages_deleted.Busi	
attribute), 33	model_computed_fields
model_computed_fields	(aiogram.types.chat_location.ChatLocation
(aiogram.types.business_opening_hours.Busine	
attribute), 33	model_computed_fields
model_computed_fields	(aiogram.types.chat_member.ChatMember
(aiogram.types.business_opening_hours_interva	
attribute), 34	model_computed_fields
model_computed_fields	(aiogram.types.chat_member_administrator.ChatMemberAdminis
(aiogram.types.callback_game.CallbackGame	attribute), 105
attribute), 364	model_computed_fields
model_computed_fields	(aiogram.types.chat_member_banned.ChatMemberBanned
(aiogram.types.callback_query.CallbackQuery	attribute), 106
attribute), 35	model_computed_fields
<pre>model_computed_fields (aiogram.types.chat.Chat at- tribute), 39</pre>	(aiogram.types.chat_member_left.ChatMemberLeft attribute), 106
model_computed_fields	model_computed_fields
	AdministratonRightsm.types.chat_member_member.ChatMemberMember
attribute), 52	attribute), 107
model_computed_fields	model_computed_fields
(aiogram.types.chat_background.ChatBackgrou	
attribute), 53	attribute), 107
model_computed_fields	model_computed_fields
(aiogram.types.chat_boost.ChatBoost at-	
tribute), 54	attribute), 108
model_computed_fields	model_computed_fields
(aiogram.types.chat_boost_added.ChatBoostAd	
attribute), 54	attribute), 122
model_computed_fields	model_computed_fields
(aiogram.types.chat_boost_removed.ChatBoosti	
attribute) 55	attrihute) 130

model_computed_fields	model_computed_fields
(aiogram.types.chat_photo.ChatPhoto at-	(aiogram.types.general_forum_topic_hidden.GeneralForumTopic
tribute), 131	attribute), 139
model_computed_fields	model_computed_fields
(aiogram.types.chat_shared.ChatShared	(aiogram.types.general_forum_topic_unhidden.GeneralForumTop
attribute), 132	attribute), 140
model_computed_fields	model_computed_fields
(aiogram.types.chosen_inline_result.ChosenInlin attribute), 284	140
model_computed_fields	model_computed_fields
(aiogram.types.contact.Contact attribute), 132	(aiogram.types.giveaway_completed.GiveawayCompleted attribute), 141
<pre>model_computed_fields (aiogram.types.dice.Dice at-</pre>	model_computed_fields
tribute), 133	(aiogram.types.giveaway_created.GiveawayCreated
model_computed_fields	attribute), 141
(aiogram.types.document.Document attribute),	
134	(aiogram.types.giveaway_winners.GiveawayWinners
model_computed_fields	attribute), 142
(aiogram.types.encrypted_credentials.Encrypted	
attribute), 350	(aiogram.types.inaccessible_message.InaccessibleMessage
model_computed_fields	attribute), 181
(aiogram.types.encrypted_passport_element.Enc	
attribute), 351	(aiogram.types.inline_keyboard_button.InlineKeyboardButton
model_computed_fields	attribute), 183
(aiogram.types.error_event.ErrorEvent at- tribute), 625	model_computed_fields
model_computed_fields	(aiogram.types.inline_keyboard_markup.InlineKeyboardMarkup attribute), 184
(aiogram.types.external_reply_info.ExternalRepl	
attribute), 135	(aiogram.types.inline_query.InlineQuery
model_computed_fields (aiogram.types.file.File at-	attribute), 285
tribute), 136	model_computed_fields
model_computed_fields	(aiogram.types.inline_query_result.InlineQueryResult
(aiogram.types.force_reply.ForceReply at-	attribute), 286
<i>tribute</i> ), 137	model_computed_fields
model_computed_fields	(aiogram.types.inline_query_result_article.InlineQueryResultArti
(aiogram.types.forum_topic.ForumTopic	attribute), 288
attribute), 138	model_computed_fields
model_computed_fields	(aiogram.types.inline_query_result_audio.InlineQueryResultAudi
(aiogram.types.forum_topic_closed.ForumTopicC	
attribute), 138	model_computed_fields
model_computed_fields	(aiogram.types.inline_query_result_cached_audio.InlineQueryRe
(aiogram.types.forum_topic_created.ForumTopic	
attribute), 138	model_computed_fields
model_computed_fields	(aiogram.types.inline_query_result_cached_document.InlineQuer
(aiogram.types.forum_topic_edited.ForumTopicE	
attribute), 139	model_computed_fields
model_computed_fields	(aiogram.types.inline_query_result_cached_gif.InlineQueryResult_result_cached_gif.InlineQueryResult_result_
(aiogram.types.forum_topic_reopened.ForumTop	•
attribute), 139	model_computed_fields
model_computed_fields (aiogram.types.game.Game attribute), 364	(aiogram.types.inline_query_result_cached_mpeg4_gif.InlineQue attribute), 298
model_computed_fields	model_computed_fields
(aiogram.types.game_high_score.GameHighScore	
attribute), 365	attribute), 300

```
model_computed_fields
                                                       model_computed_fields
         (aiogram.types.inline_query_result_cached_sticker.InlineQueniyResult@pek.edfutchenedia_animation.InputMediaAnimation
                                                                attribute), 186
         attribute), 302
model_computed_fields
                                                       model_computed_fields
         (aiogram.types.inline_query_result_cached_video.InlineQu&wjRgsawhtGypoke.ihVjideomedia_audio.InputMediaAudio
        attribute), 304
                                                                attribute), 188
model_computed_fields
                                                       model_computed_fields
         (aiogram.types.inline_query_result_cached_voice.InlineQueryiRgsultaCyphsdlypintemedia_document.InputMediaDocument
         attribute), 306
                                                                attribute), 189
model_computed_fields
                                                       model_computed_fields
         (aiogram.types.inline_query_result_contact.InlineQueryRes(di6gmtmctypes.input_media_photo.InputMediaPhoto
         attribute), 307
                                                                attribute), 190
model_computed_fields
                                                       model_computed_fields
         (aiogram.types.inline_query_result_document.InlineQueryRasiolgPanatupastinput_media_video.InputMediaVideo
         attribute), 310
                                                                attribute), 191
model_computed_fields
                                                       model_computed_fields
         (aiogram.types.inline_query_result_game.InlineQueryResult@iongwam.types.input_message_content.InputMessageContent
         attribute), 311
                                                                attribute), 330
model_computed_fields
                                                       model_computed_fields
         (aiogram.types.inline query result gif.InlineQueryResultG{faiogram.types.input paid media.InputPaidMedia
         attribute), 313
                                                                attribute), 191
model_computed_fields
                                                       model_computed_fields
         (aiogram.types.inline_query_result_location.InlineQueryRekaltkgoantitypes.input_paid_media_photo.InputPaidMediaPhoto
         attribute), 315
                                                                attribute), 192
model_computed_fields
                                                       model_computed_fields
         (aiogram.types.inline_query_result_mpeg4_gif.InlineQueryResingtMpay#Esifnput_paid_media_video.InputPaidMediaVideo
         attribute), 317
                                                                attribute), 193
model_computed_fields
                                                       model_computed_fields
         (aiogram.types.inline_query_result_photo.InlineQueryResult#ibotram.types.input_poll_option.InputPollOption
         attribute), 319
                                                                attribute), 193
model_computed_fields
                                                       model_computed_fields
         (aiogram.types.inline_query_result_venue.InlineQueryResultMergram.types.input_sticker.InputSticker
         attribute), 321
                                                                attribute), 333
model_computed_fields
                                                       model_computed_fields
         (aiogram.types.inline_query_result_video.InlineQueryResultWidgram.types.input_text_message_content.InputTextMessageCo
        attribute), 323
                                                                attribute), 331
model_computed_fields
                                                       model_computed_fields
         (aiogram.types.inline_query_result_voice.InlineQueryResult\u00e4ioigeam.types.input_venue_message_content.InputVenueMessag
         attribute), 324
                                                                attribute), 332
                                                       model_computed_fields
model_computed_fields
         (aiogram.types.inline query results button.InlineQueryRes(disQuatoutypes.invoice.Invoice
                                                                                                  attribute),
         attribute), 325
                                                                337
model_computed_fields
                                                       model_computed_fields
         (aiogram.types.input_contact_message_content.InputContacttlessamestSpesschatyboard_button.KeyboardButton
         attribute), 326
                                                                attribute), 194
model_computed_fields
                                                       model_computed_fields
         (aiogram.types.input_invoice_message_content.InputInvoiceMiessageCaputs.kteyboard_button_poll_type.KeyboardButtonPollT
        attribute), 328
                                                                attribute), 195
model_computed_fields
                                                       model_computed_fields
         (aiogram.types.input_location_message_content.InputLocat(aiddessauge@asukeyboard_button_request_chat.KeyboardButtonRo
         attribute), 330
                                                                attribute), 196
model_computed_fields
                                                       model_computed_fields
         (aiogram.types.input_media.InputMedia
                                                                (aiogram.types.keyboard_button_request_user.KeyboardButtonRe
```

attribute), 197

attribute), 185

attribute), 259

model_computed_fields	model_computed_fields KeyboardBu <b>(triongReagm.eypldsan</b> essage_origin_hidden_user.MessageOriginHidd
attribute), 198	attribute), 260
model_computed_fields	model_computed_fields
(aiogram.types.labeled_price.LabeledPrice	(aiogram.types.message_origin_user.MessageOriginUser
attribute), 337	attribute), 261
model_computed_fields	model_computed_fields
(atogram.types.tink_preview_options.LinkPrevie attribute), 199	wOptions (aiogram.types.message_reaction_count_updated.MessageReaction_attribute), 261
model_computed_fields	model_computed_fields
(aiogram.types.location.Location attribute),	(aiogram.types.message_reaction_updated.MessageReactionUpdated.Messa
200	attribute), 262
model_computed_fields	model_computed_fields
(aiogram.types.login_url.LoginUrl attribute),	(aiogram.types.order_info.OrderInfo attribute),
201	338
model_computed_fields	model_computed_fields
(aiogram.types.mask_position.MaskPosition attribute), 334	(aiogram.types.paid_media.PaidMedia at- tribute), 263
model_computed_fields	model_computed_fields
	sybeInacces(x <b>ibbg)Menstype</b> s.paid_media_info.PaidMediaInfo
attribute), 201	attribute), 263
model_computed_fields	model_computed_fields
(aiogram.types.menu_button.MenuButton	(aiogram.types.paid_media_photo.PaidMediaPhoto
attribute), 201	attribute), 263
model_computed_fields	model_computed_fields
	uttonComm( <b>aid</b> gram.types.paid_media_preview.PaidMediaPreview
attribute), 202	attribute), 264
model_computed_fields	model_computed_fields
	nDefault (aiogram.types.paid_media_purchased.PaidMediaPurchased
attribute), 202	attribute), 338
model_computed_fields	model_computed_fields
	tonWebApp(aiogram.types.paid_media_video.PaidMediaVideo
attribute), 203	attribute), 264
model_computed_fields	model_computed_fields
(aiogram.types.message.Message attribute),	(aiogram.types.passport_data.PassportData
209	attribute), 351
model_computed_fields	model_computed_fields
(aiogram.types.message_auto_delete_timer_char	nged.Messag <b>a PagraDre lypeTsippæs Gloan gele</b> ment_error.PassportElementError
attribute), 257	attribute), 352
model_computed_fields	model_computed_fields
(aiogram.types.message_entity.MessageEntity	(aiogram.types.passport_element_error_data_field.PassportElem
attribute), 257	attribute), 353
model_computed_fields	model_computed_fields
(aiogram.types.message_id.MessageId at-	(aiogram.types.passport_element_error_file.PassportElementErro
tribute), 258	attribute), 353
model_computed_fields	model_computed_fields
(aiogram.types.message_origin.MessageOrigin	(aiogram.types.passport_element_error_files.PassportElementErr
attribute), 258	attribute), 354
model_computed_fields	model_computed_fields
	eOriginChd <b>wie</b> gram.types.passport_element_error_front_side.PassportElem
attribute), 259	attribute), 355
model_computed_fields	model_computed_fields
(aiogram.types.message_origin_chat.MessageOr	iginChat (aiogram.types.passport_element_error_reverse_side.PassportEle

768 Index

attribute), 356

model_computed_fields	(aiogram.types.reply_keyboard_remove.ReplyKeyboardRemove
(aiogram.types.passport_element_error_selfie.P	
attribute), 357	model_computed_fields
model_computed_fields	(aiogram.types.reply_parameters.ReplyParameters
(aiogram.types.passport_element_error_transla	
attribute), 358	model_computed_fields
model_computed_fields	$(a iogram. types. response\_parameters. Response Parameters$
(aiogram.types.passport_element_error_transla	
attribute), 358	model_computed_fields
model_computed_fields	(aiogram.types.revenue_withdrawal_state.RevenueWithdrawalSta
(aiogram.types.passport_element_error_unspec	
attribute), 359	model_computed_fields
model_computed_fields	(aiogram.types.revenue_withdrawal_state_failed.RevenueWithdra
(aiogram.types.passport_file.PassportFile	attribute), 341
attribute), 360	model_computed_fields
model_computed_fields	(aiogram.types.revenue_withdrawal_state_pending.RevenueWithd
(aiogram.types.photo_size.PhotoSize attribute),	attribute), 341
265	model_computed_fields
model_computed_fields (aiogram.types.poll.Poll attribute), 266	(aiogram.types.revenue_withdrawal_state_succeeded.RevenueWitattribute), 342
model_computed_fields	model_computed_fields
(aiogram.types.poll_answer.PollAnswer at-	(aiogram.types.sent_web_app_message.SentWebAppMessage
tribute), 266	attribute), 333
model_computed_fields	model_computed_fields
(aiogram.types.poll_option.PollOption at-	(aiogram.types.shared_user.SharedUser
tribute), 267	attribute), 273
model_computed_fields	model_computed_fields
(aiogram.types.pre_checkout_query.PreCheckou	
attribute), 339	attribute), 343
model_computed_fields	model_computed_fields
(aiogram.types.proximity_alert_triggered.Proximattribute), 267	nityAlertTri <b>ggeogd</b> am.types.shipping_option.ShippingOption attribute), 343
model_computed_fields	model_computed_fields
(aiogram.types.reaction_count.ReactionCount	(aiogram.types.shipping_query.ShippingQuery
attribute), 268	attribute), 344
model_computed_fields	<pre>model_computed_fields</pre>
(aiogram.types.reaction_type.ReactionType	(aiogram.types.star_transaction.StarTransaction
attribute), 268	attribute), 345
model_computed_fields	model_computed_fields
(aiogram.types.reaction_type_custom_emoji.Rea	actionTypeC( <b>ustogrEmay</b> ipes.star_transactions.StarTransactions
attribute), 269	attribute), 345
model_computed_fields	<pre>model_computed_fields (aiogram.types.sticker.Sticker</pre>
(aiogram.types.reaction_type_emoji.ReactionTyp	peEmoji attribute), 335
attribute), 269	model_computed_fields
model_computed_fields	(aiogram.types.sticker_set.StickerSet attribute),
(aiogram.types.reaction_type_paid.ReactionType	ePaid 336
attribute), 269	<pre>model_computed_fields (aiogram.types.story.Story</pre>
model_computed_fields	attribute), 273
(aiogram.types.refunded_payment.RefundedPayment)	
attribute), 340	(aiogram.types.successful_payment.SuccessfulPayment
model_computed_fields	attribute), 346
(aiogram.types.reply_keyboard_markup.ReplyK	
attribute), 270	(aiogram.types.switch_inline_query_chosen_chat.SwitchInlineQu
model_computed_fields	attribute), 274

model_computed_fields	model_computed_fields (aiogram.types.voice.Voice
(aiogram.types.text_quote.TextQuote attribute),	attribute), 282
275	model_computed_fields
model_computed_fields	(aiogram.types.web_app_data.WebAppData
(aiogram.types.transaction_partner.Transaction)	
attribute), 347	model_computed_fields
model_computed_fields	(aiogram.types.web_app_info.WebAppInfo
(aiogram.types.transaction_partner_fragment.Tr attribute), 347	model_computed_fields
model_computed_fields	(aiogram.types.webhook_info.WebhookInfo
(aiogram.types.transaction_partner_other.Transa	
attribute), 348	model_computed_fields
model_computed_fields	(aiogram.types.write_access_allowed.WriteAccessAllowed
(aiogram.types.transaction_partner_telegram_ac	
attribute), 348	model_computed_fields
model_computed_fields	(aiogram.utils.web_app.WebAppChat at-
(aiogram.types.transaction_partner_user.Transa	
attribute), 349	model_computed_fields
model_computed_fields	(aiogram.utils.web_app.WebAppInitData
(aiogram.types.update.Update attribute), 361	attribute), 655
model_computed_fields (aiogram.types.user.User at-	<pre>model_computed_fields      (aiogram.utils.web_app.WebAppUser at-</pre>
tribute), 276	(aiogram.utils.web_app.WebAppUser at- tribute), 657
model_computed_fields	model_config (aiogram.utils.web_app.WebAppChat at-
(aiogram.types.user_chat_boosts.UserChatBoost	
attribute), 277	model_config (aiogram.utils.web_app.WebAppInitData
model_computed_fields	attribute), 655
	Phoodel_config (aiogram.utils.web_app.WebAppUser at-
attribute), 277	tribute), 657
model_computed_fields	model_fields (aiogram.utils.web_app.WebAppChat at-
(aiogram.types.user_shared.UserShared	tribute), 658
attribute), 278	$\verb model_fields  (a iogram.utils.web\_app.WebAppInitData $
model_computed_fields	attribute), 655
(aiogram.types.users_shared.UsersShared attribute), 278	model_fields (aiogram.utils.web_app.WebAppUser attribute), 657
model_computed_fields (aiogram.types.venue.Venue	model_post_init() (aiogram.methods.add_sticker_to_set.AddStickerTo
attribute), 279	method), 365
	<pre>model_post_init() (aiogram.methods.answer_callback_query.AnswerC</pre>
attribute), 280	method), 384
model_computed_fields	<pre>model_post_init() (aiogram.methods.answer_inline_query.AnswerInli</pre>
(aiogram.types.video_chat_ended.VideoChatEnd	
attribute), 280	<pre>model_post_init() (aiogram.methods.answer_pre_checkout_query.Ans</pre>
model_computed_fields	method), 530
$(a iogram. types. video\_chat\_participants\_invited.$	V <b>iuhdoAh@Bs.tti.dipiin.(s)/(viiteg</b> lram.methods.answer_shipping_query.AnswerS
attribute), 280	method), 532
model_computed_fields	<pre>model_post_init() (aiogram.methods.answer_web_app_query.Answer)</pre>
(aiogram.types.video_chat_scheduled.VideoChar	
attribute), 281	<pre>model_post_init() (aiogram.methods.approve_chat_join_request.Appr</pre>
model_computed_fields	method), 385
• • •	untactdel_post_init() (aiogram.methods.ban_chat_member.BanChatMem
attribute), 281	method), 387
model_computed_fields	model_post_init() (aiogram.methods.ban_chat_sender_chat.BanChatS
(aiogram.types.video_note.VideoNote at- tribute), 282	method), 388 model_post_init() (aiogram.methods.close.Close
11 toute j, 202	<pre>model_post_init() (aiogram.methods.close.Close</pre>

- method), 389 method), 515
- model\_post\_init() (aiogram.methods.close\_forum\_topicmGkbelEposnTipriit() (aiogram.methods.edit\_message\_text.EditMessageText.ed
- model\_post\_init() (aiogram.methods.close\_general\_formodelpip6stoseiGierteralFoognaffopniethods.export\_chat\_invite\_link.ExportChate\_invite\_link.ExportC
- model\_post\_init() (aiogram.methods.copy\_message.Copyddedsagest\_init() (aiogram.methods.forward\_message.ForwardMessage.method), 393 method), 413
- model\_post\_init() (aiogram.methods.copy\_messages.CappyMedssppest\_init() (aiogram.methods.forward\_messages.ForwardMessa method), 395
  method), 414
- model\_post\_init() (aiogram.methods.create\_chat\_invite\_mbdle\_CrposeCharlinu(i)e(dinkgram.methods.get\_business\_connection.GetBusin method), 396 method), 415
- model\_post\_init() (aiogram.methods.create\_chat\_subsc**mpdieli\_poste\_linkt())e(uie@frat@uhsthripits**cgeli<u>n</u>ethaLfaktChat method), 398 method), 416
- model\_post\_init() (aiogram.methods.create\_forum\_topim@deltafostumIhoipid() (aiogram.methods.get\_chat\_administrators.GetChatAmethod), 400 method), 417
- model\_post\_init() (aiogram.methods.create\_invoice\_lin**h.cdelitelos**tic**irhink**() (aiogram.methods.get\_chat\_member.GetChatMembe method), 534 method), 418
- model\_post\_init() (aiogram.methods.decline\_chat\_join\_modelstploecditieClta(t)d(niRgquuestnethods.get\_chat\_menu\_button.GetChatM method), 401 method), 420
- model\_post\_init() (aiogram.methods.delete\_chat\_photorNodele@flossPhintit() (aiogram.methods.get\_custom\_emoji\_stickers.GetCumethod), 402 method), 370
- model\_post\_init() (aiogram.methods.delete\_chat\_stickemodeDepersChinSttcker&eiogram.methods.get\_file.GetFile method), 403 method), 421
- model\_post\_init() (aiogram.methods.delete\_forum\_topi**mbled**te**:fostmTkpir**() (aiogram.methods.get\_forum\_topic\_icon\_stickers.Ge method), 404 method), 422
- model\_post\_init() (aiogram.methods.delete\_message.Dalleddelete\_spoget\_init() (aiogram.methods.get\_game\_high\_scores.GetGameFinethod), 507

  method), 526
- model\_post\_init() (aiogram.methods.delete\_messages.DmbdeMeysosgesinit() (aiogram.methods.get\_me.GetMe method), 508 method), 423
- model\_post\_init() (aiogram.methods.delete\_my\_comma**mdsl&lelepost**lyGovintn@n(ksiogram.methods.get\_my\_commands.GetMyCommands, delete\_my\_commands.GetMyCommands, delete\_my\_commands.GetMyCommands.GetMyCommands, delete\_my\_commands.GetMyCommands.
- model\_post\_init() (aiogram.methods.delete\_sticker\_from\_ode\_D\_plexStirkint()) (msogram.methods.get\_my\_default\_administrator\_rig method), 368 method), 425
- model\_post\_init() (aiogram.methods.delete\_sticker\_set.foodsdSpickstSinit() (aiogram.methods.get\_my\_description.GetMyDescrip method), 369

  method), 426

  model\_post\_init() (aiogram.methods.get\_my\_description.GetMyDescrip
- model\_post\_init() (aiogram.methods.edit\_chat\_invite\_limbdedit\_chbsstlniinidtif() (aiogram.methods.get\_my\_short\_description.GetMysmethod), 407

  method), 428
- model\_post\_init() (aiogram.methods.edit\_chat\_subscrip**niode\_In:pioes\_ti:riknEiti() (haii&ghser:ipeithoHsvigetListk**r\_transactions.GetStarTransaction), 408

  method), 535
- model\_post\_init() (aiogram.methods.edit\_forum\_topic.EndidEdr\_upoExp\_icnit() (aiogram.methods.get\_sticker\_set.GetStickerSet method), 409 method), 371
- model\_post\_init() (aiogram.methods.edit\_general\_forumodoplc.piditGeineitalforumfapic.methods.get\_updates.GetUpdates method), 410 method), 542
- model\_post\_init() (aiogram.methods.edit\_message\_captiond EditAsssaign Cupioniogram.methods.get\_user\_chat\_boosts.GetUserChamethod), 510

  method), 429
- model\_post\_init() (aiogram.methods.edit\_message\_live\_mlodaliopdEstit/tilnsstage\_Liviogram.tinathods.get\_user\_profile\_photos.GetUserI method), 512 method), 429
- model\_post\_init() (aiogram.methods.edit\_message\_medinodelit\_MessugitMed() (aiogram.methods.get\_webhook\_info.GetWebhookInj method), 514 method), 543
- model\_post\_init() (aiogram.methods.edit message repl**modelk.ppo.EditMiss**(**)**eRindwMurkenhods.hide general forum topic.HideG

- method), 430
  method), 477
  model\_post\_init() (aiogram.methods.leave\_chat.LeaveChadel\_post\_init() (aiogram.methods.send\_video\_note.SendVideoNote method), 431
  model\_post\_init() (aiogram.methods.log\_out.LogOut model\_post\_init() (aiogram.methods.send\_voice.SendVoice method), 432
  model\_post\_init() (aiogram.methods.pin\_chat\_messagenRidehanVastsaigrit() (aiogram.methods.set\_chat\_administrator\_custom\_ti
- method), 433

  method), 484

  model\_post\_init() (aiogram.methods.promote\_chat\_membereProposteChaitMe)n(lociogram.methods.set\_chat\_description.SetChatDesc
- method), 436

  method), 485

  model post init() (aiogram methods refund star paym**mustel** Afund Star Privilent (aiogram methods set chat menu button Set Chat M
- model\_post\_init() (aiogram.methods.refund\_star\_paym**enode**dfu**pdSt**ar**Pnyw()**n(aiogram.methods.set\_chat\_menu\_button.SetChatMe method), 536 method), 486 model\_post\_init() (aiogram.methods.reopen\_forum\_top**ixoRedp.podStr\_uinflop()**) (aiogram.methods.set\_chat\_permissions.SetChatPern
- method), 437 method), 488
  model\_post\_init() (aiogram.methods.reopen\_general\_formathe\_logicoste\_openicte) method), 488
- model\_post\_init() (aiogram.methods.reopen\_general\_formordie\_topposte\_open\_telepleatibgrammilelpluods.set\_chat\_photo.SetChatPhoto method), 438 method), 489 model\_post\_init() (aiogram.methods.replace\_sticker\_in<u>modeRepl</u>ocseStickivtKi)Seniogram.methods.set\_chat\_sticker\_set.SetChatSticke
- method), 372 method), 490
  model\_post\_init() (aiogram.methods.restrict\_chat\_membodRestpicsChittlitenibeniogram.methods.set\_chat\_title.SetChatTitle
- method), 439 method), 491
- model\_post\_init() (aiogram.methods.revoke\_chat\_invitembik!ReposteChrilm()) (blingram.methods.set\_custom\_emoji\_sticker\_set\_thumethod), 441 method), 375
- model\_post\_init() (aiogram.methods.send\_animation.Samokleiimproism\_init() (aiogram.methods.set\_game\_score.SetGameScore method), 443 method), 529
- model\_post\_init() (aiogram.methods.send\_chat\_action.shodkUhaptAstioimit() (aiogram.methods.set\_my\_commands.SetMyComman method), 448 method), 494
- model\_post\_init() (aiogram.methods.send\_contact.Sendmodel\_post\_init() (aiogram.methods.set\_my\_default\_administrator\_rig method), 450 method), 495 model\_post\_init() (aiogram.methods.send\_dice.SendDiamodel\_post\_init() (aiogram.methods.set\_my\_description.SetMyDescrip
- method), 452 method), 496 model\_post\_init() (aiogram.methods.send\_document.SendDel:upræst\_init() (aiogram.methods.set\_my\_name.SetMyName
- model\_post\_init() (aiogram.methods.send\_document.Se**ndDelupast**\_init() (aiogram.methods.set\_my\_name.SetMyName method), 455 method), 497
- method), 455

  model\_post\_init() (aiogram.methods.send\_game.SendGmodel\_post\_init() (aiogram.methods.set\_my\_short\_description.SetMyS

  method), 498
- model\_post\_init() (aiogram.methods.send\_invoice.SendModede\_post\_init() (aiogram.methods.set\_passport\_data\_errors.SetPass method), 539
  method), 546
- model\_post\_init() (aiogram.methods.send\_location.Senathodentiapost\_init() (aiogram.methods.set\_sticker\_emoji\_list.SetStickerEn
  method), 457
  method), 376
- model\_post\_init() (aiogram.methods.send\_media\_groupm**SchelMpdistGini**pt() (aiogram.methods.set\_sticker\_keywords.SetStickerKe method), 460 method), 377
- model\_post\_init() (aiogram.methods.send\_message.Sen**nlbdtes**tagost\_init() (aiogram.methods.set\_sticker\_mask\_position.SetS
- model\_post\_init() (aiogram.methods.send\_paid\_media.**SeodiffaipbMetlia**init() (aiogram.methods.set\_sticker\_position\_in\_set.SetStic method), 465 method), 379
- model\_post\_init() (aiogram.methods.send\_photo.SendPhotdel\_post\_init() (aiogram.methods.set\_sticker\_set\_thumbnail.SetSticker\_method), 468
  method), 380
- model\_post\_init() (aiogram.methods.send\_poll.SendPollmodel\_post\_init() (aiogram.methods.set\_sticker\_set\_title.SetStickerSetTimethod), 471
  method), 382
- model\_post\_init() (aiogram.methods.send\_venue.SendVmodel\_post\_init() (aiogram.methods.stop\_message\_live\_location.StopM method), 474
  method), 519
- $\verb|model_post_init()| (a iogram.methods.send\_video.SendVinked el\_post_init()| (a iogram.methods.stop\_poll.StopPoll)| (a iogram.methods.stop\_poll.StopPoll.StopPoll)| (a iogram.methods.stop\_poll.StopPoll.Stop$

*method*), 520 method), 29 model\_post\_init() (aiogram.methods.unban\_chat\_memb**road&ibapaStatMein**k&r(aiogram.types.bot\_command\_scope\_chat\_member.B method), 499 method), 30 model\_post\_init() (aiogram.methods.unban\_chat\_sendemodleth\_bloksanGhattSendartGhattSendartGhatt.types.bot\_command\_scope\_default.BotCom method), 500 method), 30 model\_post\_init() (aiogram.methods.unhide general formorde logiostyliidis general formorde log method), 502 method), 30 model\_post\_init() (aiogram.methods.unpin\_all\_chat\_maxodeds\_bastmAllChatMaxogram.types.bot\_name.BotName method), 502 method), 31 model\_post\_init() (aiogram.methods.unpin\_all\_forum\_mpinelmpssstesib/inp(n)Allfagram/loppieMasssstasrt\_description.BotShortDescr method), 504 method), 31 model\_post\_init() (aiogram.methods.unpin\_all\_generalmodaln\_porsit\_inacista@ekalibprionAll@eesetrukhoesen\_doppineMieseadkusinessConnec method), 505 method), 32 model\_post\_init() (aiogram.methods.unpin\_chat\_messa**read&lpip6&txt/Maix**@(aiogram.types.business\_intro.BusinessIntro method), 32 method), 506 model\_post\_init() (aiogram.methods.upload\_sticker\_filemb@beladsiskeiffile() (aiogram.types.business\_location.BusinessLocation method), 383 method), 32 model\_post\_init() (aiogram.types.animation.Animation model\_post\_init() (aiogram.types.business\_messages\_deleted.Business. method), 20 method), 33 model\_post\_init() (aiogram.types.audio.Audio model\_post\_init() (aiogram.types.business\_opening\_hours.BusinessOp method), 20 method), 33 model\_post\_init() (aiogram.types.background\_fill.Background\_fillba method), 34 method), 21 model\_post\_init() (aiogram.types.background fill freefmodedraphiestr.BackgroundfillfmeefpermGiltlidieht game.CallbackGame method), 21 method), 364 model\_post\_init() (aiogram.types.background\_fill\_gradievod&hcppstundfillGradiogram.types.callback\_query.CallbackQuery method), 22 method), 35 model\_post\_init() (aiogram.types.background\_fill\_solidnicatekrppostEilhSibtid) (aiogram.types.chat.Chat method), method), 22 model\_post\_init() (aiogram.types.background\_type.Background\_type.lack\_init() (aiogram.types.chat\_administrator\_rights.ChatAdmi method), 23 method), 53 model\_post\_init() (aiogram.types.background\_type\_chanodenephackground\_fynic@natifilugnes.chat\_background.ChatBackground method), 23 method), 53 model\_post\_init() (aiogram.types.background\_type\_fill.**RadkgropnatTypatitl**() (aiogram.types.chat\_boost.ChatBoost method), 24 method), 54  $model\_post\_init()$  (aiogram.types.background\_type\\_pattwodsHaqdgsauridsTypePuniargram.types.chat\_boost\_added.ChatBoostAdded method), 24 method), 54 model\_post\_init() (aiogram.types.background\_type\_walkhoodper\_Roodkground\_type@Workhoodperpes.chat\_boost\_removed.ChatBoostRemo method), 25 method), 55 model\_post\_init() (aiogram.types.birthdate.Birthdate model\_post\_init() (aiogram.types.chat\_boost\_source.ChatBoostSource method), 25 method), 55 model\_post\_init() (aiogram.types.bot\_command.BotCommondmeld\_post\_init() (aiogram.types.chat\_boost\_source\_gift\_code.ChatBo method), 26 method), 56 model\_post\_init() (aiogram.types.bot\_command\_scope.**RodGomprostdStrapte**() (aiogram.types.chat\_boost\_source\_giveaway.ChatBo method), 26 method), 57 model\_post\_init() (aiogram.types.bot\_command\_scope\_madlechaposdnninistedip(siBgtGoontupersdSaapbAddChatAchei.poistmainors.ChatBo method), 27 method), 57 model\_post\_init() (aiogram.types.bot\_command\_scope\_mdlerppsthaitsiBo(ConingradiStreppesAdlarobopOshattpdated.ChatBoostUpdated method), 27 method), 57 model\_post\_init() (aiogram.types.bot\_command\_scope\_modestripost\_chiati.B6) CoingrammalScoperAllHrfudite@fileushatFullInfo method), 28 method), 60 model\_post\_init() (aiogram.types.bot\_command\_scope\_modelbeapGotupianidScopedGhatm.types.chat\_invite\_link.ChatInviteLink

Index 773

model\_post\_init() (aiogram.types.bot command scope rubidelaphoististiation.(BatGogmanutudsescopleChiatAndmeiquiestatGhrstJoinRequest

method), 62

method), 28

model\_post\_init() (aiogram.types.chat\_location.ChatLomotlent\_post\_init()

method), 96

*method*), 103

method), 103

method), 105

method), 106

*method*), 106

method), 107

method), 107

method), 109

model\_post\_init() (aiogram.types.chat\_member\_updaterik@altMposkerikipda@daiogram.types.inaccessible\_message.InaccessibleM method), 122 method), 181 model\_post\_init() (aiogram.types.chat\_permissions.ChantBetrikispiasts\_init() (aiogram.types.inline\_keyboard\_button.InlineKeyboa method), 130 method), 183 model\_post\_init() (aiogram.types.chat\_photo.ChatPhotomodel\_post\_init() (aiogram.types.inline\_keyboard\_markup.InlineKeybo method), 184 method), 131 model\_post\_init() (aiogram.types.chat shared.ChatShareotel\_post\_init() (aiogram.types.inline query.InlineQuery *method*), 132 method), 285 model\_post\_init() (aiogram.types.chosen\_inline\_result.@bdernIplistRixnilt() (aiogram.types.inline\_query\_result.InlineQueryResu *method*), 284 method), 287 model\_post\_init() (aiogram.types.contact.Contact model\_post\_init() (aiogram.types.inline\_query\_result\_article.InlineQu *method*), 132 method), 288 model\_post\_init() (aiogram.types.dice.Dice method), model\_post\_init() (aiogram.types.inline\_query\_result\_audio.InlineQue method), 289 model\_post\_init() (aiogram.types.document.Document model\_post\_init() (aiogram.types.inline\_query\_result\_cached\_audio.In method), 134 method), 292 model\_post\_init() (aiogram.types.encrypted\_credential.mbde.lypped:Credeviti()s(aiogram.types.inline\_query\_result\_cached\_docume. *method*), 350 method), 294 model\_post\_init() (aiogram.types.encrypted\_passport\_alerdeht\_fostypiedRasspoidfilamanpes.inline\_query\_result\_cached\_gif.Inlin method), 351 method), 296 model\_post\_init() (aiogram.types.error\_event.ErrorEvemodel\_post\_init() (aiogram.types.inline\_query\_result\_cached\_mpeg4\_ method), 625 method), 298 model\_post\_init() (aiogram.types.external\_reply\_info.Emtedal[Papstlnfoit() (aiogram.types.inline\_query\_result\_cached\_photo.In method), 300 *method*), 135 model\_post\_init() (aiogram.types.file.File method), model\_post\_init() (aiogram.types.inline\_query\_result\_cached\_sticker.I. method), 302  $model\_post\_init()$  (aiogram.types.force\_reply.ForceReplyodel\\_post\\_init() (aiogram.types.inline\_query\_result\_cached\_video.In method), 137 method), 304 model\_post\_init() (aiogram.types.forum\_topic.ForumTopicdel\_post\_init() (aiogram.types.inline\_query\_result\_cached\_voice.In method), 138 method), 306 model\_post\_init() (aiogram.types.forum\_topic\_closed.FnodreTopicsClosedt() (aiogram.types.inline\_query\_result\_contact.InlineQu *method*), 138 method), 307 model\_post\_init() (aiogram.types.forum\_topic\_created.floodenlTopoisCriantetl() (aiogram.types.inline\_query\_result\_document.Inline method), 310 *method*), 138 model\_post\_init() (aiogram.types.forum\_topic\_edited.FandarTopioEtlitrdit() (aiogram.types.inline\_query\_result\_game.InlineQuer method), 139 method), 311 model\_post\_init() (aiogram.types.forum topic reopeneato Governs Lawye (adaiogram.types.inline query result gif.Inline Query I

method), 139

method), 364

method), 365

method), 139

method), 140

method), 140

method), 141

method), 141

method), 142

model\_post\_init() (aiogram.types.chat\_member.ChatMemodeel\_post\_init() (aiogram.types.game\_high\_score.GameHighScore

model\_post\_init() (aiogram.types.chat member admini**nrodxir.@locstM.einbtr(Administratio**rpes.general forum topic hidden.Genera

model\_post\_init() (aiogram.types.chat\_member\_bannedm**6talMpostre\_Bnittet)** (aiogram.types.general\_forum\_topic\_unhidden.Gene

model\_post\_init() (aiogram.types.chat\_member\_membe**m.cda:lyMporsh**er**iMiorub)** (aiogram.types.giveaway\_completed.GiveawayComp

model\_post\_init() (aiogram.types.chat\_member\_owner.@hdeWepdstQimiet() (aiogram.types.giveaway\_created.GiveawayCreated

model\_post\_init() (aiogram.types.chat\_member\_restric**mdd&latMest**hb**irtles(t)idteid**gram.types.giveaway\_winners.GiveawayWinners

model\_post\_init() (aiogram.types.chat\_member\_left.Chmmdehbpokaft\_init() (aiogram.types.giveaway.Giveaway

(aiogram.types.game.Game

method), 195 *method*), 313 model\_post\_init() (aiogram.types.inline\_query\_result\_lmodtidn\_poliste@mirtyRekalthgoantitypes.keyboard\_button\_request\_chat.Keybo method), 315 method), 196 model\_post\_init() (aiogram.types.inline\_query\_result\_model\_gipbstlipieQue();ReindtMpay#Esikeyboard\_button\_request\_user.Keybo method), 317 method), 197 model\_post\_init() (aiogram.types.inline\_query\_result\_**nhodeInlinesQuixniRe**()**ultdFilogna**m.types.keyboard\_button\_request\_users.Keyb method), 198 method), 319 model\_post\_init() (aiogram.types.inline\_query\_result\_vande\_Infines@uarniResultiMergram.types.labeled\_price.LabeledPrice method), 321 method), 337 model\_post\_init() (aiogram.types.inline\_query\_result\_windade.ln|posQueinyRe(i)|(Windgram.types.link\_preview\_options.LinkPreviewOp method), 323 method), 199 model\_post\_init() (aiogram.types.inline\_query\_result\_wnodeln!postQueingResQut\mathbb{vaiogram.types.location.Location method), 324 method), 200 model\_post\_init() (aiogram.types.inline\_query\_results\_hottel.lplistQimit(Res(disk)ununtypes.login\_url.LoginUrl method), 201 method), 325 model\_post\_init() (aiogram.types.input\_contact\_messagnodeltepros.str.pitGotQuctVitegramgeGpesemtask\_position.MaskPosition *method*), 326 method), 334 model\_post\_init() (aiogram.types.input\_invoice\_messagaodentepostruithiva()eMessagaaCypasmtaybe\_inaccessible\_message.Maybe *method*), 328 method), 201 model\_post\_init() (aiogram.types.input\_location\_messagededniposkupintlx&dt(aiddeasagefeasagefeasagenu\_button.MenuButton method), 330 method), 202 model\_post\_init() (aiogram.types.input\_media.InputMemiadel\_post\_init() (aiogram.types.menu\_button\_commands.MenuButton method), 202 method), 185 model\_post\_init() (aiogram.types.input media animati**moderlupbosdig@mirt@ioa**iogram.types.menu button default.MenuButtonDe *method*), 186 method), 202 model\_post\_init() (aiogram.types.input\_media\_audio.Impodled.jpostd.jpnit() (aiogram.types.menu\_button\_web\_app.MenuButtonV method), 188 method), 203 model\_post\_init() (aiogram.types.input\_media\_docume**molisplutMest**ialDnitwf)ent(aiogram.types.message.Message method), 189 method), 209 model\_post\_init() (aiogram.types.input\_media\_photo.Impodled.photStotionit() (aiogram.types.message\_auto\_delete\_timer\_changed *method*), 190 method), 257 model\_post\_init() (aiogram.types.input\_media\_video.Inpadlelkdpd&tleanit() (aiogram.types.message\_entity.MessageEntity method), 191 method), 257 model\_post\_init() (aiogram.types.input\_message\_contemologlutydesstasiaGartiontaiogram.types.message\_id.MessageId *method*), 330 method), 258 model\_post\_init() (aiogram.types.input\_paid\_media.Inp**nofleid!Medit\_**init() (aiogram.types.message\_origin.MessageOrigin method), 191 method), 258 model\_post\_init() (aiogram.types.input\_paid\_media\_ph**nool.bip.ppbstd\_Maiogram**.types.message\_origin\_channel.MessageOr. method), 192 method), 259 model\_post\_init() (aiogram.types.input\_paid\_media\_vi**otaodkilp.ppBs.td/Media(V)deio**ogram.types.message\_origin\_chat.MessageOrigin method), 193 method), 260 model\_post\_init() (aiogram.types.input\_poll\_option.Inp**model/Qptish**\_init() (aiogram.types.message\_origin\_hidden\_user.Messag method), 193 method), 260 model\_post\_init() (aiogram.types.input\_sticker.InputSticMardel\_post\_init() (aiogram.types.message\_origin\_user.MessageOrigin *method*), 333 method), 261 model\_post\_init() (aiogram.types.input\_text\_message\_coordealt\_lppstTeixt\delta\square\gamma\text\_ending\text\_postTeixt\delta\square\text\_postTeixt\delta\tex

method), 194 method), 263
model\_post\_init()(aiogram.types.keyboard\_button\_pollmoytelKpykstariddSyten.types.paid\_media\_info.PaidMediaInfo

model\_post\_init() (aiogram.types.input\_venue\_messagemodrele\_potreput\_VarieteMessageCounteprets.message\_reaction\_updated.Messagel

method), 261

method), 262

method), 338

(aiogram.types.invoice.Invoice model\_post\_init() (aiogram.types.order\_info.OrderInfo

Index 775

model\_post\_init() (aiogram.types.keyboard\_button.KeyhnodedBuptost\_init() (aiogram.types.paid\_media.PaidMedia

*method*), 331

*method*), 332

*method*), 337

model\_post\_init()

method), 268

method), 268

*method*), 269

*method*), 269

*method*), 263 method), 269 model\_post\_init() (aiogram.types.paid\_media\_photo.PaindMediapHsoto\_init() (aiogram.types.refunded\_payment.RefundedPayment method), 263 method), 340 model\_post\_init() (aiogram.types.paid\_media\_preview.Roidle1eqfior\$reviewt() (aiogram.types.reply\_keyboard\_markup.ReplyKeybo method), 264 method), 270 model\_post\_init() (aiogram.types.paid media purchasadbeell/MeditaPimalta(sedaiogram.types.reply keyboard remove.ReplyKeyboa method), 271 method), 338 model\_post\_init() (aiogram.types.paid\_media\_video.Paiabdediabidediabideto\_init() (aiogram.types.reply\_parameters.ReplyParameters method), 264 method), 272 model\_post\_init() (aiogram.types.passport\_data.Passpo**rttdet**h\_post\_init() (aiogram.types.response\_parameters.ResponseParam method), 351 method), 272 model\_post\_init() (aiogram.types.passport\_element\_ermodPeils:pporsElementErfoniogram.types.revenue\_withdrawal\_state.RevenueW *method*), 352 method), 341 model\_post\_init() (aiogram.types.passport\_element\_ermoodedtapfiskd\_Pasisp(): (LilenneantErvjoe:DaeteeFineld\_withdrawal\_state\_failed.Rev method), 341 method), 353 model\_post\_init() (aiogram.types.passport\_element\_ermoodfele\_**Rossp\_ointElevientele**pes.revenue\_withdrawal\_state\_pending.F *method*), 353 method), 342 model\_post\_init() (aiogram.types.passport\_element\_ermoodfelesprostportEte(n) (nati Egran Fityses.revenue\_withdrawal\_state\_succeeded *method*), 354 method), 342 model\_post\_init() (aiogram.types.passport\_element\_ermodfeduposide\_Paisp@/tEleguantEnyporF.sontSiekb\_app\_message.SentWebApp *method*), 355 method), 333 model\_post\_init() (aiogram.types.passport\_element\_ermoder\represtidenRassphontElements\represtateSharedUser method), 273 method), 356model\_post\_init() (aiogram.types.passport element ermodelfip.datssiariElementSuffer.shipping address.ShippingAddress method), 357 *method*), 343 model\_post\_init() (aiogram.types.passport\_element\_ermoderlunplationifilet.Rds(spioghibhentepetExhlapPhings)laphionifile(Rd) method), 358 method), 343 model\_post\_init() (aiogram.types.passport\_element\_ermodterlupplatsionifiles(PalasipogntEhetype method), 358 method), 344 model\_post\_init() (aiogram.types.passport\_element\_ermooderlsppostle.diRistsp)rttEberrentEyprostlasperaifisedction.StarTransaction method), 359 method), 345  $model\_post\_init()$  (aiogram.types.passport\_file.Passporttideel\\_post\\_init() (aiogram.types.star\\_transactions.StarTransactions *method*), 360 method), 346 model\_post\_init() (aiogram.types.photo\_size.PhotoSize model\_post\_init() (aiogram.types.sticker.Sticker method), 265 method), 335 model\_post\_init() (aiogram.types.poll.Poll method), model\_post\_init() (aiogram.types.sticker\_set.StickerSet method), 336 model\_post\_init() (aiogram.types.poll\_answer.PollAnswmodel\_post\_init() (aiogram.types.story.Story method), 266 method), 273 model\_post\_init() (aiogram.types.poll\_option.PollOptiomodel\_post\_init() (aiogram.types.successful\_payment.SuccessfulPayme method), 267 method), 346 model\_post\_init() (aiogram.types.pre\_checkout\_query.Rued&bc|poustQuaryt() (aiogram.types.switch\_inline\_query\_chosen\_chat.Sw method), 274 *method*), 339 model\_post\_init() (aiogram.types.proximity\_alert\_triggmodelroporattyMertTriggeogdam.types.text\_quote.TextQuote method), 267 method), 275 model\_post\_init() (aiogram.types.reaction\_count.Reaction@clupost\_init() (aiogram.types.transaction\_partner.TransactionPartner.

model\_post\_init() (aiogram.types.reaction\_type\_paid.Rmodialn\_TyppsPaidhit() (aiogram.types.transaction\_partner\_user.Transaction

model\_post\_init() (aiogram.types.reaction\_type.Reactiomodel\_post\_init() (aiogram.types.transaction\_partner\_fragment.Transa

model\_post\_init() (aiogram.types.reaction\_type\_custom<u>medseli\_flosationflypeC(astogrEmaji</u>pes.transaction\_partner\_other.Transactio

model\_post\_init() (aiogram.types.reaction\_type\_emoji.lkeadstlonflyptEinvjit() (aiogram.types.transaction\_partner\_telegram\_ads.Tr

method), 347

method), 347

method), 348

method), 348

```
554
             method), 349
model_post_init()
                                   (aiogram.types.update.Update
                                                                                      aiogram.enums.inline_query_result_type,
            method), 361
model_post_init() (aiogram.types.user.User method),
                                                                                      aiogram.enums.input_media_type, 556
                                                                                      aiogram.enums.input_paid_media_type, 556
model_post_init() (aiogram.types.user chat boosts.UserChatRogstam.enums.keyboard_button_poll_type_type,
             method), 277
model_post_init() (aiogram.types.user_profile_photos.UserPaiofblgrPalmteenums.mask_position_point, 557
             method), 277
                                                                                      aiogram.enums.menu_button_type, 557
model_post_init() (aiogram.types.user_shared.UserShared
                                                                                     aiogram.enums.message_entity_type, 557
             method), 278
                                                                                      aiogram.enums.message_origin_type, 558
model_post_init() (aiogram.types.users_shared.UsersSharedaiogram.enums.paid_media_type, 558
             method), 278
                                                                                      aiogram.enums.parse_mode, 559
model_post_init()
                                                                                      aiogram.enums.passport_element_error_type,
                                       (aiogram.types.venue.Venue
             method), 279
model_post_init()
                                       (aiogram.types.video.Video
                                                                                      aiogram.enums.poll_type, 559
                                                                                      aiogram.enums.reaction_type_type, 560
            method), 280
model_post_init() (aiogram.types.video_chat_ended.VideoClaiv@ppdaml.enums.revenue_withdrawal_state_type,
            method), 280
model_post_init() (aiogram.types.video chat participants inaiteatr\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\divident\dividentd\divident\dividentd\divident\dividentd\divident\dividentd\dividentd\dividentd\dividentd\dividentd\dividentd\dividentd\dividentd\dividentd\dividentd\dividentd\dividentd\dividentd\dividentd\dividentd\dividentd\dividentd\dividentd\dividentd\dividentd\dividentd\dividentd\dividentd\dividentd\dividentd\dividentd\dividentd\dividentd\dividentd\dividentd\dividentd\dividentd\dividentd\dividentd\dividentd\dividentd\dividentd\dividentd\dividentd\dividentd\dividentd\dividentd\dividentd\dividentd\divide
             method), 281
                                                                                      aiogram.enums.sticker_type, 561
model_post_init() (aiogram.types.video_chat_scheduled.VidexiGlgrtSmhedrums.topic_icon_color, 561
                                                                                      aiogram.enums.transaction_partner_type,
             method), 281
model_post_init() (aiogram.types.video chat started.VideoChatStarted
                                                                                      aiogram.enums.update_type, 562
            method), 281
model_post_init() (aiogram.types.video_note.VideoNote
                                                                                      aiogram.exceptions, 626
             method), 282
                                                                                      aiogram.handlers.callback_query, 639
model_post_init()
                                                                                      aiogram.methods.add_sticker_to_set, 365
                                        (aiogram.types.voice.Voice
                                                                                      aiogram.methods.answer_callback_query,
            method), 282
model_post_init() (aiogram.types.web_app_data.WebAppData
                                                                                            384
             method), 283
                                                                                      aiogram.methods.answer_inline_query, 521
model_post_init() (aiogram.types.web_app_info.WebAppInfoaiogram.methods.answer_pre_checkout_query,
             method), 283
model_post_init() (aiogram.types.webhook_info.WebhookInfa.iogram.methods.answer_shipping_query,
             method), 363
model_post_init() (aiogram.types.write_access_allowed.Writa:AlogaxamAlfmentahods.answer_web_app_query, 524
            method), 284
                                                                                      aiogram.methods.approve_chat_join_request,
model_post_init() (aiogram.utils.web_app.WebAppChat
             method), 658
                                                                                      aiogram.methods.ban_chat_member, 386
model_post_init() (aiogram.utils.web_app.WebAppInitData aiogram.methods.ban_chat_sender_chat, 388
                                                                                      aiogram.methods.close, 389
            method), 655
model_post_init() (aiogram.utils.web_app.WebAppUser
                                                                                      aiogram.methods.close_forum_topic, 390
                                                                                      aiogram.methods.close_general_forum_topic,
             method), 657
module
                                                                                             391
      aiogram.dispatcher.flags, 628
                                                                                      aiogram.methods.copy_message, 392
      aiogram.enums.bot_command_scope_type, 547
                                                                                      aiogram.methods.copy_messages, 394
                                                                                      aiogram.methods.create_chat_invite_link,
      aiogram.enums.chat_action, 547
      aiogram.enums.chat_boost_source_type, 548
      aiogram.enums.chat_member_status, 548
                                                                                      aiogram.methods.create_chat_subscription_invite_link,
      aiogram.enums.chat_type, 549
      aiogram.enums.content_type, 549
                                                                                      aiogram.methods.create_forum_topic, 399
                                                                                      aiogram.methods.create_invoice_link,533
      aiogram.enums.currency, 551
      aiogram.enums.dice_emoji, 554
                                                                                      aiogram.methods.create_new_sticker_set,
      aiogram.enums.encrypted_passport_element,
                                                                                             366
```

```
aiogram.methods.decline_chat_join_request,
                                               aiogram.methods.get_star_transactions,
    400
                                                   535
aiogram.methods.delete_chat_photo, 402
                                               aiogram.methods.get_sticker_set, 371
aiogram.methods.delete_chat_sticker_set,
                                               aiogram.methods.get_updates,541
                                               aiogram.methods.get_user_chat_boosts, 428
aiogram.methods.delete_forum_topic, 404
                                               aiogram.methods.get_user_profile_photos,
aiogram.methods.delete_message, 507
aiogram.methods.delete_messages, 508
                                               aiogram.methods.get_webhook_info,543
aiogram.methods.delete_my_commands, 405
                                               aiogram.methods.hide_general_forum_topic,
aiogram.methods.delete_sticker_from_set,
                                               aiogram.methods.leave_chat, 431
aiogram.methods.delete_sticker_set, 369
                                               aiogram.methods.log_out, 432
aiogram.methods.delete_webhook, 540
                                               aiogram.methods.pin_chat_message, 433
aiogram.methods.edit_chat_invite_link,
                                               aiogram.methods.promote_chat_member, 434
                                               aiogram.methods.refund_star_payment,536
aiogram.methods.edit_chat_subscription_invite_dimmdram.methods.reopen_forum_topic, 437
    408
                                               aiogram.methods.reopen_general_forum_topic,
aiogram.methods.edit_forum_topic, 409
aiogram.methods.edit_general_forum_topic,
                                               aiogram.methods.replace_sticker_in_set,
aiogram.methods.edit_message_caption, 509
                                               aiogram.methods.restrict_chat_member, 439
aiogram.methods.edit_message_live_location,
                                               aiogram.methods.revoke_chat_invite_link,
    511
                                                   440
aiogram.methods.edit_message_media,513
                                               aiogram.methods.send_animation, 442
aiogram.methods.edit_message_reply_markup,
                                               aiogram.methods.send_audio, 445
                                               aiogram.methods.send_chat_action, 447
aiogram.methods.edit_message_text,516
                                               aiogram.methods.send_contact, 449
aiogram.methods.export_chat_invite_link,
                                               aiogram.methods.send_dice, 451
                                               aiogram.methods.send_document, 453
aiogram.methods.forward_message, 412
                                               aiogram.methods.send_game, 526
aiogram.methods.forward_messages, 414
                                               aiogram.methods.send_invoice, 537
aiogram.methods.get_business_connection,
                                               aiogram.methods.send_location, 456
                                               aiogram.methods.send_media_group, 459
                                               aiogram.methods.send_message, 461
aiogram.methods.get_chat, 416
aiogram.methods.get_chat_administrators,
                                               aiogram.methods.send_paid_media,464
                                               aiogram.methods.send_photo, 466
aiogram.methods.get_chat_member, 418
                                               aiogram.methods.send_poll, 469
aiogram.methods.get_chat_member_count,
                                               aiogram.methods.send_sticker, 373
    419
                                               aiogram.methods.send_venue, 473
                                               aiogram.methods.send_video, 475
aiogram.methods.get_chat_menu_button, 420
aiogram.methods.get_custom_emoji_stickers,
                                               aiogram.methods.send_video_note, 479
    370
                                               aiogram.methods.send_voice, 481
aiogram.methods.get_file, 421
                                               aiogram.methods.set_chat_administrator_custom_title,
aiogram.methods.get_forum_topic_icon_stickers,
                                                   484
                                               aiogram.methods.set_chat_description, 485
aiogram.methods.get_game_high_scores, 525
                                               aiogram.methods.set_chat_menu_button, 486
aiogram.methods.get_me, 423
                                               aiogram.methods.set_chat_permissions, 487
                                               aiogram.methods.set_chat_photo, 489
aiogram.methods.get_my_commands, 424
aiogram.methods.get_my_default_administrator_raighgram.methods.set_chat_sticker_set,490
                                               aiogram.methods.set_chat_title, 491
aiogram.methods.get_my_description, 426
                                               aiogram.methods.set_custom_emoji_sticker_set_thumbnail
aiogram.methods.get_my_name, 427
aiogram.methods.get_my_short_description,
                                               aiogram.methods.set_game_score, 529
    428
                                               aiogram.methods.set_message_reaction, 492
```

```
aiogram.methods.set_my_commands, 493
                                               aiogram.types.bot_command_scope, 26
aiogram.methods.set_my_default_administrator_raighgram.types.bot_command_scope_all_chat_administrator
aiogram.methods.set_my_description, 496
                                               aiogram.types.bot_command_scope_all_group_chats,
aiogram.methods.set_my_name, 497
aiogram.methods.set_my_short_description,
                                               aiogram.types.bot_command_scope_all_private_chats,
aiogram.methods.set_passport_data_errors,
                                               aiogram.types.bot_command_scope_chat, 28
                                               aiogram.types.bot_command_scope_chat_administrators,
aiogram.methods.set_sticker_emoji_list,
                                               aiogram.types.bot_command_scope_chat_member,
aiogram.methods.set_sticker_keywords, 377
aiogram.methods.set_sticker_mask_position,
                                               aiogram.types.bot_command_scope_default,
aiogram.methods.set_sticker_position_in_set,
                                               aiogram.types.bot_description, 30
                                               aiogram.types.bot_name, 31
                                               aiogram.types.bot_short_description, 31
aiogram.methods.set_sticker_set_thumbnail,
                                               aiogram.types.business_connection, 31
                                               aiogram.types.business_intro,32
aiogram.methods.set_sticker_set_title,
                                               aiogram.types.business_location, 32
    382
aiogram.methods.set_webhook, 543
                                               aiogram.types.business_messages_deleted,
aiogram.methods.stop_message_live_location,
    518
                                               aiogram.types.business_opening_hours, 33
aiogram.methods.stop_poll,520
                                               aiogram.types.business_opening_hours_interval,
                                                   34
aiogram.methods.unban_chat_member, 499
aiogram.methods.unban_chat_sender_chat,
                                               aiogram.types.callback_game, 364
                                               aiogram.types.callback_query, 34
aiogram.methods.unhide_general_forum_topic,
                                               aiogram.types.chat, 36
                                               aiogram.types.chat_administrator_rights,
aiogram.methods.unpin_all_chat_messages,
                                               aiogram.types.chat_background, 53
aiogram.methods.unpin_all_forum_topic_messagesaiogram.types.chat_boost,54
                                               aiogram.types.chat_boost_added, 54
aiogram.methods.unpin_all_general_forum_topic_amicsspaaguestypes.chat_boost_removed, 55
                                               aiogram.types.chat_boost_source, 55
aiogram.methods.unpin_chat_message, 505
                                               aiogram.types.chat_boost_source_gift_code,
aiogram.methods.upload_sticker_file, 383
aiogram.types.animation, 19
                                               aiogram.types.chat_boost_source_giveaway,
aiogram.types.audio, 20
aiogram.types.background_fill, 21
                                               aiogram.types.chat_boost_source_premium,
aiogram.types.background_fill_freeform_gradient, 57
                                               aiogram.types.chat_boost_updated, 57
                                               aiogram.types.chat_full_info,58
aiogram.types.background_fill_gradient,
                                               aiogram.types.chat_invite_link,61
aiogram.types.background_fill_solid, 22
                                               aiogram.types.chat_join_request,62
aiogram.types.background_type, 23
                                               aiogram.types.chat_location, 103
aiogram.types.background_type_chat_theme,
                                               aiogram.types.chat_member, 103
                                               aiogram.types.chat_member_administrator,
aiogram.types.background_type_fill, 23
                                               aiogram.types.chat_member_banned, 106
aiogram.types.background_type_pattern, 24
aiogram.types.background_type_wallpaper,
                                               aiogram.types.chat_member_left, 106
                                               aiogram.types.chat_member_member, 107
aiogram.types.birthdate, 25
                                               aiogram.types.chat_member_owner, 107
aiogram.types.bot_command, 26
                                               aiogram.types.chat_member_restricted, 108
```

```
aiogram.types.chat_member_updated, 109
                                               aiogram.types.inline_query_result_cached_voice,
aiogram.types.chat_permissions, 130
aiogram.types.chat_photo, 131
                                               aiogram.types.inline_query_result_contact,
aiogram.types.chat_shared, 132
aiogram.types.chosen_inline_result, 284
                                               aiogram.types.inline_query_result_document,
aiogram.types.contact, 132
aiogram.types.dice, 133
                                               aiogram.types.inline_query_result_game,
aiogram.types.document, 133
aiogram.types.encrypted_credentials, 349
                                               aiogram.types.inline_query_result_gif,
aiogram.types.encrypted_passport_element,
    350
                                               aiogram.types.inline_query_result_location,
aiogram.types.error_event, 625
aiogram.types.external_reply_info, 134
                                               aiogram.types.inline_query_result_mpeg4_gif,
aiogram.types.file, 136
                                                   315
aiogram.types.force_reply, 137
                                               aiogram.types.inline_query_result_photo,
aiogram.types.forum_topic, 137
aiogram.types.forum_topic_closed, 138
                                               aiogram.types.inline_query_result_venue,
aiogram.types.forum_topic_created, 138
aiogram.types.forum_topic_edited, 139
                                               aiogram.types.inline_query_result_video,
aiogram.types.forum_topic_reopened, 139
aiogram.types.game, 364
                                               aiogram.types.inline_query_result_voice,
aiogram.types.game_high_score, 365
aiogram.types.general_forum_topic_hidden,
                                               aiogram.types.inline_query_results_button,
aiogram.types.general_forum_topic_unhidden,
                                               aiogram.types.input_contact_message_content,
aiogram.types.giveaway, 140
                                               aiogram.types.input_file, 184
aiogram.types.giveaway_completed, 141
                                               aiogram.types.input_invoice_message_content,
aiogram.types.giveaway_created, 141
aiogram.types.giveaway_winners, 142
                                               aiogram.types.input_location_message_content,
aiogram.types.inaccessible_message, 143
aiogram.types.inline_keyboard_button, 183
                                               aiogram.types.input_media, 185
                                               aiogram.types.input_media_animation, 185
aiogram.types.inline_keyboard_markup, 184
aiogram.types.inline_query, 284
                                               aiogram.types.input_media_audio, 187
aiogram.types.inline_query_result, 286
                                               aiogram.types.input_media_document, 188
aiogram.types.inline_query_result_article,
                                               aiogram.types.input_media_photo, 189
                                               aiogram.types.input_media_video, 190
aiogram.types.inline_query_result_audio,
                                               aiogram.types.input_message_content, 330
                                               aiogram.types.input_paid_media, 191
aiogram.types.inline_query_result_cached_audioaiogram.types.input_paid_media_photo, 192
                                               aiogram.types.input_paid_media_video, 192
aiogram.types.inline_query_result_cached_documaeintgram.types.input_poll_option, 193
                                               aiogram.types.input_sticker, 333
aiogram.types.inline_query_result_cached_gif, aiogram.types.input_text_message_content,
aiogram.types.inline_query_result_cached_mpeg4aipigram.types.input_venue_message_content,
aiogram.types.inline_query_result_cached_photoaiogram.types.invoice, 337
                                               aiogram.types.keyboard_button, 194
aiogram.types.inline_query_result_cached_stickaringram.types.keyboard_button_poll_type,
aiogram.types.inline_query_result_cached_videoaiogram.types.keyboard_button_request_chat,
    302
                                               aiogram.types.keyboard_button_request_user,
```

```
197
                                               aiogram.types.passport_element_error_unspecified,
aiogram.types.keyboard_button_request_users,
                                                    359
                                               aiogram.types.passport_file, 360
aiogram.types.labeled_price, 337
                                               aiogram.types.photo_size, 265
aiogram.types.link_preview_options, 199
                                               aiogram.types.poll, 265
aiogram.types.location, 200
                                               aiogram.types.poll_answer, 266
aiogram.types.login_url, 200
                                               aiogram.types.poll_option, 267
aiogram.types.mask_position, 334
                                               aiogram.types.pre_checkout_query, 339
aiogram.types.maybe_inaccessible_message,
                                               aiogram.types.proximity_alert_triggered,
aiogram.types.menu_button, 201
                                               aiogram.types.reaction_count, 268
aiogram.types.menu_button_commands, 202
                                               aiogram.types.reaction_type, 268
aiogram.types.menu_button_default, 202
                                               aiogram.types.reaction_type_custom_emoji,
aiogram.types.menu_button_web_app, 203
                                                    268
                                               aiogram.types.reaction_type_emoji, 269
aiogram.types.message, 203
aiogram.types.message_auto_delete_timer_changeatiogram.types.reaction_type_paid, 269
    257
                                               aiogram.types.refunded_payment, 340
aiogram.types.message_entity, 257
                                               aiogram.types.reply_keyboard_markup, 270
aiogram.types.message_id, 258
                                               aiogram.types.reply_keyboard_remove, 271
aiogram.types.message_origin, 258
                                               aiogram.types.reply_parameters, 271
aiogram.types.message_origin_channel, 259
                                               aiogram.types.response_parameters, 272
aiogram.types.message_origin_chat, 259
                                               aiogram.types.revenue_withdrawal_state,
aiogram.types.message_origin_hidden_user,
                                               aiogram.types.revenue_withdrawal_state_failed,
aiogram.types.message_origin_user, 260
aiogram.types.message_reaction_count_updated, aiogram.types.revenue_withdrawal_state_pending,
                                               aiogram.types.revenue_withdrawal_state_succeeded,
aiogram.types.message_reaction_updated,
    262
aiogram.types.order_info, 338
                                               aiogram.types.sent_web_app_message, 333
aiogram.types.paid_media, 263
                                               aiogram.types.shared_user, 273
aiogram.types.paid_media_info, 263
                                               aiogram.types.shipping_address, 343
aiogram.types.paid_media_photo, 263
                                               aiogram.types.shipping_option, 343
aiogram.types.paid_media_preview, 264
                                               aiogram.types.shipping_query, 344
aiogram.types.paid_media_purchased, 338
                                               aiogram.types.star_transaction, 345
aiogram.types.paid_media_video, 264
                                               aiogram.types.star_transactions, 345
aiogram.types.passport_data, 351
                                               aiogram.types.sticker, 334
aiogram.types.passport_element_error, 352
                                               aiogram.types.sticker_set, 336
aiogram.types.passport_element_error_data_fieladiogram.types.story, 273
                                               aiogram.types.successful_payment, 346
    352
aiogram.types.passport_element_error_file,
                                               aiogram.types.switch_inline_query_chosen_chat,
    353
                                                    274
aiogram.types.passport_element_error_files,
                                               aiogram.types.text_quote, 275
    354
                                               aiogram.types.transaction_partner, 347
aiogram.types.passport_element_error_front_sidam.ogram.types.transaction_partner_fragment,
    354
                                                    347
aiogram.types.passport_element_error_reverse_saimgram.types.transaction_partner_other,
                                                   348
aiogram.types.passport_element_error_selfie,
                                               aiogram.types.transaction_partner_telegram_ads,
aiogram.types.passport_element_error_translatiaoiogfidme, types.transaction_partner_user,
\verb|aiogram.types.passport_element_error_translatia \verb|aiogram.types.update|, 360|
    358
                                               aiogram.types.user, 275
```

```
aiogram.types.user_chat_boosts, 277
                                                                                                                                            name (aiogram.methods.create new sticker set.CreateNewStickerSet
           aiogram.types.user_profile_photos, 277
                                                                                                                                                                   attribute), 366
           aiogram.types.user_shared, 277
                                                                                                                                            name (aiogram.methods.delete sticker set.DeleteStickerSet
           aiogram.types.users_shared, 278
                                                                                                                                                                   attribute), 369
           aiogram.types.venue, 279
                                                                                                                                            {\tt name}\ (aiogram.methods.edit\_chat\_invite\_link.EditChatInviteLink
           aiogram.types.video, 279
                                                                                                                                                                   attribute), 406
           aiogram.types.video_chat_ended, 280
                                                                                                                                            name (aiogram.methods.edit chat subscription invite link.EditChatSubscr
           aiogram.types.video_chat_participants_invited,
                                                                                                                                                                    attribute), 408
                       280
                                                                                                                                            name (aiogram.methods.edit forum topic.EditForumTopic
           aiogram.types.video_chat_scheduled, 281
                                                                                                                                                                   attribute), 409
           aiogram.types.video_chat_started, 281
                                                                                                                                            \verb"name" (a iogram.methods.edit\_general\_forum\_topic.EditGeneralForumTopic) \\
           aiogram.types.video_note, 281
                                                                                                                                                                    attribute), 410
           aiogram.types.voice, 282
                                                                                                                                            name (aiogram.methods.get_sticker_set.GetStickerSet at-
           aiogram.types.web_app_data, 283
                                                                                                                                                                   tribute), 371
           aiogram.types.web_app_info,283
                                                                                                                                            {\tt name}\,(aiogram.methods.replace\_sticker\_in\_set.ReplaceStickerInSet
           aiogram.types.webhook_info, 363
                                                                                                                                                                   attribute), 372
           aiogram.types.write_access_allowed, 283
                                                                                                                                            name (aiogram.methods.set_custom_emoji_sticker_set_thumbnail.SetCustor
MongoStorage (class in aiogram.fsm.storage.mongo),
                                                                                                                                                                   attribute), 375
                                                                                                                                                                          (aiogram.methods.set_my_name.SetMyName
                                                                                                                                            name
month (aiogram.types.birthdate.Birthdate attribute), 25
                                                                                                                                                                    attribute), 497
MOUTH (aiogram.enums.mask_position_point.MaskPositionProdute (aiogram.methods.set_sticker_set_thumbnail.SetStickerSetThumbnail
                      attribute), 557
                                                                                                                                                                   attribute), 380
mpeg4_duration(aiogram.types.inline_query_result_mpegs4_meif(hibiga@uerstReadstMpegi4&eif_set_title.SetStickerSetTitle
                       attribute), 317
                                                                                                                                                                   attribute), 382
mpeg4_file_id(aiogram.types.inline_query_result_cachedn_ampege4_gifutm.tippeQubatyResultBatNeadMpatp4bitfe), 31
                       attribute), 298
                                                                                                                                            name (aiogram.types.chat invite link.ChatInviteLink at-
MPEG4_GIF (aiogram.enums.inline_query_result_type.InlineQueryResultitype, 62
                                                                                                                                            name (aiogram.types.forum_topic.ForumTopic attribute),
                       attribute), 555
mpeg4_height (aiogram.types.inline_query_result_mpeg4_gif.Inline@ideryResultMpeg4Gif
                       attribute), 317
                                                                                                                                            name (aiogram.types.forum_topic_created.ForumTopicCreated
mpeg4_url (aiogram.types.inline_query_result_mpeg4_gif.InlineQueryRebuteMpleg4Gif
                       attribute), 317
                                                                                                                                            name\ (aiogram.types.forum\_topic\_edited.ForumTopicEdited
mpeg4_width(aiogram.types.inline_query_result_mpeg4_gif.InlineQutriblexa)ltMpeg4Gif
                                                                                                                                            name (aiogram.types.order_info.OrderInfo attribute), 338
                       attribute), 317
MUR (aiogram.enums.currency.Currency attribute), 553
                                                                                                                                            name (aiogram.types.sticker set.StickerSet attribute), 336
MVR (aiogram.enums.currency.Currency attribute), 553
                                                                                                                                            {\tt need\_email}\ (aiogram.methods.create\_invoice\_link.CreateInvoiceLink
MXN (aiogram.enums.currency.Currency attribute), 553
                                                                                                                                                                   attribute), 534
MY_CHAT_MEMBER (aiogram.enums.update_type.UpdateTypeneed_email (aiogram.methods.send_invoice.SendInvoice
                       attribute), 562
                                                                                                                                                                    attribute), 539
my_chat_member
                                               (aiogram.types.update.Update
                                                                                                                                           need_email(aiogram.types.input_invoice_message_content.InputInvoiceM
                       tribute), 362
                                                                                                                                                                   attribute), 329
                                                                                                                                            {\tt need\_name} \ (aiogram.methods.create\_invoice\_link.CreateInvoiceLink
MYR (aiogram.enums.currency.Currency attribute), 553
MZN (aiogram.enums.currency.Currency attribute), 553
                                                                                                                                                                   attribute), 534
                                                                                                                                            need\_name\ (aiogram.methods.send\_invoice.SendInvoice
Ν
                                                                                                                                                                   attribute), 539
\verb|name|| (a iogram.methods.add\_sticker\_to\_set.AddStickerToSet| \verb|need\_name|| (a iogram.types.input\_invoice\_message\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessage\_content.InputInvoiceMessa
                                                                                                                                                                   attribute), 328
                       attribute), 365
{\tt name} \ (aiogram.methods.create\_chat\_invite\_link.CreateChat \verb| Market | Dhone\_number (aiogram.methods.create\_invoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.CreateInvoice\_link.
                                                                                                                                                                    attribute), 534
                       attribute), 396
name (aiogram.methods.create_chat_subscription_invite_linkeedechenatshunderblioingkamennethods.send_invoice.SendInvoice
                                                                                                                                                                    attribute), 539
                       attribute), 399
{\tt name} \ (aiogram.methods.create\_forum\_topic.CreateForumToppeed\_phone\_number} \ (aiogram.types.input\_invoice\_message\_content.Input\_topic.CreateForumToppeed\_phone\_number) \ (aiogram.types.input\_topic.CreateForumToppeed\_phone\_number) \ (aiogram.types.types.types.types.types.types.types
                                                                                                                                                                    attribute), 328
                       attribute), 399
                                                                                                                                            need_shipping_address
```

(aiogram.methods.create_invoice_link.CreateInvatribute), 534	oicled_indeaction(aiogram.types.message_reaction_updated.MessageReaction_ attribute), 262
need_shipping_address	old_sticker(aiogram.methods.replace_sticker_in_set.ReplaceStickerInS
(aiogram.methods.send_invoice.SendInvoice	attribute), 372
attribute), 539	one_time_keyboard(aiogram.types.reply_keyboard_markup.ReplyKeybo
need_shipping_address	attribute), 270
	n <b>pnlly_o</b> if <b>.elv!assag</b> d@i <b>ogeat</b> n.methods.unban_chat_member.UnbanChatMen
attribute), 329	attribute), 499
needs_repainting (aiogram.methods.create_new_sticker	
attribute), 367	attribute), 140
	only_new_members (aiogram.types.giveaway_winners.GiveawayWinners
attribute), 335	attribute), 142
new_chat_member(aiogram.types.chat_member_updated.	
attribute), 110	tribute), 471
NEW_CHAT_MEMBERS (aiogram.enums.content_type.Content	
attribute), 550	opening_hours (aiogram.types.business_opening_hours.BusinessOpening
new_chat_members (aiogram.types.message.Message	attribute), 33
attribute), 207	opening_minute(aiogram.types.business_opening_hours_interval.Busine
NEW_CHAT_PHOTO (aiogram.enums.content_type.ContentTy	
attribute), 550	option_ids (aiogram.types.poll_answer.PollAnswer at-
new_chat_photo (aiogram.types.message.Message at-	tribute), 266
tribute), 207	
NEW_CHAT_TITLE (aiogram.enums.content_type.ContentTy	- · · · · · · · · · · · · · · · · · · ·
attribute), 550	options (aiogram.types.poll.Poll attribute), 265
	order_info(aiogram.types.pre_checkout_query.PreCheckoutQuery
tribute), 207	attribute), 339
	MedangeRefotioidgpalates/pes.successful_payment.SuccessfulPayment
attribute), 262	attribute), 346
next_offset (aiogram.methods.answer_inline_query.Ans	
attribute), 523	origin(aiogram.types.external_reply_info.ExternalReplyInfo
NGN (aiogram.enums.currency.Currency attribute), 553	attribute), 135
NIO (aiogram.enums.currency.Currency attribute), 553	OTHER (aiogram.enums.transaction_partner_type.TransactionPartnerType
NOK (aiogram.enums.currency.Currency attribute), 553	attribute), 561
NPR (aiogram.enums.currency.Currency attribute), 553 NPR (aiogram.enums.currency.Currency attribute), 553	auriouie), 501
NZD (aiogram.enums.currency.Currency attribute), 553	P
NZD (diogram.enums.currency.Currency diiriodie), 333	
0	PAB (aiogram.enums.currency.Currency attribute), 553
	pack() (aiogram.filters.callback_data.CallbackData
${\tt offset} (a iogram.methods.get\_star\_transactions.GetStarTr$	
attribute), 535	PAID_MEDIA (aiogram.enums.content_type.ContentType
offset (aiogram.methods.get_updates.GetUpdates at-	attribute), 549
tribute), 541	paid_media(aiogram.types.external_reply_info.ExternalReplyInfo
$\verb"offset" (a iogram.methods.get\_user\_profile\_photos.GetUse") \\$	· ·
attribute), 429	paid_media (aiogram.types.message.Message attribute),
offset (aiogram.types.inline_query.InlineQuery at-	206
tribute), 285	paid_media(aiogram.types.paid_media_info.PaidMediaInfo
offset (aiogram.types.message_entity.MessageEntity at-	attribute), 263
tribute), 257	$\verb"paid_media" (a iogram. types. transaction\_partner\_user. Transaction Partner\_user. Transactio$
ok (aiogram.methods.answer_pre_checkout_query.Answer)	
attribute), 530	paid_media_payload(aiogram.types.paid_media_purchased.PaidMedia.
ok (aiogram.methods.answer_shipping_query.AnswerShipp	
attribute), 531	$\verb"paid_media_payload" (a iogram. types. transaction\_partner\_user. Transaction\_user. $
$\verb"old_chat_member" (a iogram. types. chat_member\_up dated.$	
attribute), 110	PaidMedia (class in aiogram.types.paid_media), 263

```
PaidMediaInfo
                                                                              (class
                                                                                                                                           parse_mode(aiogram.types.inline_query_result_gif.InlineQueryResultGif
                       aiogram.types.paid_media_info), 263
                                                                                                                                                                   attribute), 313
PaidMediaPhoto
                                                                                                                                           parse_mode(aiogram.types.inline query result mpeg4 gif.InlineQueryRe
                                                                              (class
                                                                                                                                 in
                       aiogram.types.paid_media_photo), 263
                                                                                                                                                                   attribute), 317
PaidMediaPreview
                                                                                  (class
                                                                                                                                 in
                                                                                                                                           parse_mode (aiogram.types.inline_query_result_photo.InlineQueryResult)
                      aiogram.types.paid media preview), 264
                                                                                                                                                                   attribute), 319
PaidMediaPurchased
                                                                                     (class
                                                                                                                                           parse_mode(aiogram.types.inline query result video.InlineQueryResultV
                                                                                                                                 in
                       aiogram.types.paid_media_purchased), 338
                                                                                                                                                                   attribute), 323
PaidMediaType
                                                                              (class
                                                                                                                                           parse_mode (aiogram.types.inline_query_result_voice.InlineQueryResultV
                                                                                                                                 in
                       aiogram.enums.paid_media_type), 558
                                                                                                                                                                   attribute), 325
PaidMediaVideo
                                                                              (class
                                                                                                                                           \verb"parse_mode" (a iogram.types.input\_media\_animation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAnimation.InputMediaAn
                                                                                                                                 in
                       aiogram.types.paid_media_video), 264
                                                                                                                                                                   attribute), 186
parse_mode(aiogram.client.default.DefaultBotProperties parse_mode(aiogram.types.input_media_audio.InputMediaAudio
                      attribute), 567
                                                                                                                                                                   attribute), 188
parse_mode(aiogram.methods.copy_message.CopyMessagparse_mode(aiogram.types.input_media_document.InputMediaDocument
                       attribute), 393
                                                                                                                                                                   attribute), 189
parse_mode (aiogram.methods.edit_message_caption.Edit/MesssageManheibaiogram.types.input_media_photo.InputMediaPhoto
                       attribute), 510
                                                                                                                                                                   attribute), 190
parse_mode (aiogram.methods.edit_message_text.EditMesspar Eextmode (aiogram.types.input_media_video.InputMediaVideo
                       attribute), 517
                                                                                                                                                                   attribute), 191
parse_mode (aiogram.methods.send_animation.SendAnimapianse_mode (aiogram.types.input_text_message_content.InputTextMessage
                       attribute), 443
                                                                                                                                                                   attribute), 331
parse_mode (aiogram.methods.send_audio.SendAudio parse_webapp_init_data()
                                                                                                                                                                                                                                       (in
                                                                                                                                                                                                                                                                module
                       attribute), 445
                                                                                                                                                                   aiogram.utils.web app), 654
parse_mode (aiogram.methods.send_document.SendDocum@arse_mode (class in aiogram.enums.parse_mode), 559
                       attribute), 454
                                                                                                                                           PASSPORT (aiogram.enums.encrypted_passport_element.EncryptedPasspor
parse_mode (aiogram.methods.send_message.SendMessage
                                                                                                                                                                   attribute), 554
                                                                                                                                           PASSPORT_DATA (aiogram.enums.content_type.ContentType
                       attribute), 462
                                                                                                                                                                  attribute), 550
parse_mode (aiogram.methods.send_paid_media.SendPaidMedia
                       attribute), 465
                                                                                                                                           passport_data
                                                                                                                                                                                                    (aiogram.types.message.Message
parse_mode
                                    (aiogram.methods.send_photo.SendPhoto
                                                                                                                                                                   attribute), 209
                       attribute), 467
                                                                                                                                           PASSPORT_REGISTRATION
                                     (aiogram.methods.send_video.SendVideo
parse_mode
                                                                                                                                                                   (aiogram.enums.encrypted_passport_element.EncryptedPassport
                       attribute), 477
                                                                                                                                                                   attribute), 555
parse_mode
                                      (aiogram.methods.send_voice.SendVoice PassportData (class in aiogram.types.passport_data),
                      attribute), 482
parse_mode(aiogram.types.inline query result audio.InliRe@sportAesluententerror
                                                                                                                                                                                                                                     (class
                                                                                                                                                                                                                                                                             in
                       attribute), 289
                                                                                                                                                                   aiogram.types.passport_element_error),
parse_mode(aiogram.types.inline_query_result_cached_audio.InlineQueryResultCachedAudio
                                                                                                                                           PassportElementErrorDataField
                       attribute), 292
parse_mode(aiogram.types.inline_query_result_cached_document.Inling@www.tyResuphwad@lonuenenerror_data_field),
                       attribute), 294
                                                                                                                                                                   352
parse_mode (aiogram.types.inline_query_result_cached_gift insigned it Ethern to the cached_gift in the cached gift in the cache
                                                                                                                                                                                                                                          (class
                                                                                                                                                                                                                                                                            in
                       attribute), 296
                                                                                                                                                                   aiogram.types.passport_element_error_file),
parse_mode(aiogram.types.inline_query_result_cached_mpeg4_gif.InfineQueryResultCachedMpeg4Gif
                                                                                                                                           PassportElementErrorFiles
                       attribute), 298
                                                                                                                                                                                                                                            (class
                                                                                                                                                                                                                                                                             in
parse_mode (aiogram.types.inline_query_result_cached_photo.Inline@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.yde@agraphes.ydegaagraphes.ydegaagraphes.ydegaagraphes.ydegaagraphes.ydeg
                       attribute), 300
                                                                                                                                                                   354
parse_mode(aiogram.types.inline_query_result_cached_villas.spbra@lemeResEdttConFredWt&cde
                                                                                                                                                                                                                                                                             in
                       attribute), 304
                                                                                                                                                                   aiogram.types.passport_element_error_front_side),
parse_mode(aiogram.types.inline_query_result_cached_voice.Inline@weryResultCachedVoice
                                                                                                                                           PassportElementErrorReverseSide
                      attribute), 306
parse_mode (aiogram.types.inline_query_result_document.InlineQuearidRessult.Dypessrpenssport_element_error_reverse_side),
                      attribute), 310
                                                                                                                                                                   355
```

```
PassportElementErrorSelfie
                                                           (class
                                                                                                   attribute), 59
              aiogram.types.passport_element_error_selfie),
                                                                                     PERSONAL_DETAILS (aiogram.enums.encrypted_passport_element.Encrypt
PassportElementErrorTranslationFile (class in PHONE_NUMBER(aiogram.enums.encrypted_passport_element.EncryptedPa
             aiogram.types.passport_element_error_translation_file),
                                                                                                   attribute), 555
                                                                                     {\tt PHONE\_NUMBER}\,(aiogram.enums.message\_entity\_type.MessageEntityType
PassportElementErrorTranslationFiles (class in
                                                                                                   attribute), 557
              aiogram.types.passport_element_error_translatiophidae,,number (aiogram.methods.send_contact.SendContact
              358
                                                                                                   attribute), 449
PassportElementErrorType
                                                                                    phone_number
                                                                                                                (aiogram.types.contact.Contact
                                                         (class
                                                                               in
              aiogram.enums.passport_element_error_type),
                                                                                                   tribute), 132
                                                                                     phone_number (aiogram.types.encrypted_passport_element.EncryptedPass
PassportElementErrorUnspecified
                                                                (class
                                                                                                   attribute), 350
                                                                              in
             aiogram.types.passport_element_error_unspecifierflyone_number (aiogram.types.inline_query_result_contact.InlineQueryRe
                                                                                                   attribute), 307
PassportFile (class in aiogram.types.passport_file), phone_number(aiogram.types.input_contact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_content.InputContact_message_c
                                                                                                   attribute), 326
pattern (aiogram.filters.exception.ExceptionMessageFilterphone_number (aiogram.types.order_info.OrderInfo at-
              attribute), 589
                                                                                                   tribute), 338
pay (aiogram.types.inline_keyboard_button.InlineKeyboardBhaneNumber (class in aiogram.utils.formatting), 666
              attribute), 184
                                                                                     PHOTO
                                                                                                  (aiogram.enums.content_type.ContentType at-
payload(aiogram.methods.create_invoice_link.CreateInvoiceLink
                                                                                                  tribute), 549
                                                                                     PHOTO (aiogram.enums.inline_query_result_type.InlineQueryResultType
              attribute), 533
payload (aiogram.methods.send_invoice.SendInvoice at-
                                                                                                   attribute), 555
                                                                                     {\tt PHOTO}\,(aiogram.enums.input\_media\_type.InputMediaType
              tribute), 538
payload(aiogram.methods.send_paid_media.SendPaidMedia
                                                                                                   attribute), 556
              attribute), 465
                                                                                     {\tt PHOTO}\,(aiogram.enums.input\_paid\_media\_type.InputPaidMediaType
payload (aiogram.types.input_invoice_message_content.InputInvoiceMessburge)Content
                                                                                     PHOTO (aiogram.enums.paid_media_type.PaidMediaType
             attribute), 328
PEN (aiogram.enums.currency.Currency attribute), 553
                                                                                                   attribute), 558
PENDING (aiogram.enums.revenue_withdrawal_state_type.Replacture WilladogwallStateHoods.send_photo.SendPhoto
              attribute), 560
                                                                                                   tribute), 467
                                                                                                  (aiogram.methods.set_chat_photo.SetChatPhoto
pending_join_request_count
                                                                                     photo
              (aiogram.types.chat_invite_link.ChatInviteLink
                                                                                                   attribute), 489
             attribute), 62
                                                                                     photo (aiogram.types.chat.Chat attribute), 39
pending_update_count
                                                                                     photo
                                                                                                 (aiogram.types.chat_full_info.ChatFullInfo
              (aiogram.types.webhook_info.WebhookInfo
                                                                                                   tribute), 59
              attribute), 363
                                                                                     photo
                                                                                                    (aiogram.types.chat_shared.ChatShared
performer (aiogram.methods.send_audio.SendAudio at-
                                                                                                   tribute), 132
             tribute), 446
                                                                                     photo(aiogram.types.external_reply_info.ExternalReplyInfo
performer (aiogram.types.audio.Audio attribute), 20
                                                                                                   attribute), 135
performer (aiogram.types.inline_query_result_audio.Inline@actyReingltAmdigpes.game.Game attribute), 364
             attribute), 290
                                                                                     photo (aiogram.types.message.Message attribute), 206
performer (aiogram.types.input_media_audio.InputMedia.phdito (aiogram.types.paid_media_photo.PaidMediaPhoto
             attribute), 188
                                                                                                   attribute), 264
permissions (aiogram.methods.restrict_chat_member.RestphachatMemibgram.types.shared_user.SharedUser
              attribute), 439
                                                                                                   tribute), 273
permissions (aiogram.methods.set_chat_permissions.SetChhooRornfixlionid (aiogram.types.inline_query_result_cached_photo.InlineQ
              attribute), 487
                                                                                                   attribute), 300
permissions (aiogram.types.chat.Chat attribute), 39
                                                                                     \verb|photo_height| (a iogram.methods.create_invoice\_link.CreateInvoiceLink\\
permissions(aiogram.types.chat_full_info.ChatFullInfo
                                                                                                   attribute), 534
             attribute), 60
                                                                                     \verb|photo_height| (a iogram.methods.send_invoice.SendInvoice|
personal_chat (aiogram.types.chat.Chat attribute), 39
                                                                                                   attribute), 539
```

personal\_chat (aiogram.types.chat full info.ChatFullInfaphoto\_height (aiogram.types.inline query result photo.InlineQueryResu

	attribute), 319	C	attribute), 136
photo_height(aiogram.types.input_invoice_message_conpol_lnuitlgraniceylessage&attribute), 207			
	attribute), 328	poll (aiog	gram.types.update.Update attribute), 362
photo_s	${ t size}\ (aiogram.methods.create\_invoice\_link.Create$	Prodiké <i>klak</i>	ks in aiogram.types.poll), 265
	attribute), 534	POLL_ANS	WER (aiogram.enums.update_type.UpdateType
photo_s	size (aiogram.methods.send_invoice.SendInvoice		attribute), 562
	attribute), 538	poll_ans	wer (aiogram.types.update.Update attribute),
photo_s	${ t size}(aiogram.types.input\_invoice\_message\_conte$	nt.InputInve	Blow Message Content
	attribute), 328		(aiogram.types.poll_answer.PollAnswer at-
photo_u	arl (aiogram.methods.create_invoice_link.CreateI		
	attribute), 534		er (class in aiogram.types.poll_answer), 266
photo_u	arl (aiogram.methods.send_invoice.SendInvoice		
	attribute), 538		(class in aiogram.enums.poll_type), 559
photo_u			<b>(UtilDhotton.</b> methods.set_sticker_position_in_set.SetStickerPosition
attribute), 318 attribute), 379			
photo_u	Ir1 (aiogram.types.input_invoice_message_conten attribute), 328	_	deMogrsunga Grestegaume_high_score.GameHighScore attribute), 365
photo_u	arl (aiogram.utils.web_app.WebAppChat at-	position	(aiogram.types.text_quote.TextQuote at-
_	tribute), 658		tribute), 275
photo_u	arl (aiogram.utils.web_app.WebAppUser at- tribute), 657		e (aiogram.types.shipping_address.ShippingAddress attribute), 343
photo_w			nums.message_entity_type.MessageEntityType
. –	attribute), 534		attribute), 558
photo_w	width(aiogram.methods.send_invoice.SendInvoice		
_	attribute), 538		KOUT_QUERY (aiogram.enums.update_type.UpdateType
photo_w	idth(aiogram.types.inline_query_result_photo.Ir	ılineQuery <b>k</b>	RetsribuPh)pt562
	attribute), 319	pre_chec	kout_query (aiogram.types.update.Update at-
photo_w	$\verb ridth  (a iogram.types.input\_invoice\_message\_continuous) $	ent.InputInt	wibadMe3dageContent
	attribute), 328	_	kout_query_id
photos	(aiogram.types.user_profile_photos.UserProfilePhototibute), 277		(aiogram.methods.answer_pre_checkout_query.AnswerPreCheck attribute), 530
PhotoSi	ze (class in aiogram.types.photo_size), 265	PreCheck	
PHP (aiogram.enums.currency.Currency attribute), 553 aiogram.types.pre_checkout_query), 339			
	niogram.types.message.Message method), 253	prefer_l	arge_media(aiogram.types.link_preview_options.LinkPreviewC
	ssage() (aiogram.types.chat.Chat method), 46		attribute), 199
PinChat	Message (class in	prefer_s	mall_media(aiogram.types.link_preview_options.LinkPreviewC
	aiogram.methods.pin_chat_message), 433	C	attribute), 199
PINNED_	MESSAGE (aiogram.enums.content_type.ContentTy attribute), 550		aiogram.filters.command.CommandObject at- tribute), 580
pinned_	- · · · · · · · · · · · · · · · · · · ·		aiogram.enums.chat_boost_source_type.ChatBoostSourceType
	40		attribute), 548
pinned_	message (aiogram.types.chat_full_info.ChatFullIi attribute), 60	-	animation (aiogram.types.sticker.Sticker at- tribute), 335
pinned_	message (aiogram.types.message.Message at-		subscription_month_count
	tribute), 208		(aiogram.types.giveaway.Giveaway attribute),
PKR (aio	gram.enums.currency.Currency attribute), 553	1	141
PLN (aio	gram.enums.currency.Currency attribute), 553	premium_	subscription_month_count
point	(aiogram.types.mask_position.MaskPosition attribute), 334		(aiogram.types.giveaway_winners.GiveawayWinners attribute), 142
POLL	(aiogram.enums.content_type.ContentType at- tribute), 549	prepare_	value() (aiogram.client.session.base.BaseSession nethod), 16
POLL	(aiogram.enums.update_type.UpdateType at-		nemoa), 10 aiogram.enums.paid_media_type.PaidMediaType
	tribute), 562	C	attribute), 558
$\verb poll  (a iogram. types. external\_reply\_info. External ReplyInfo \verb prices  (a iogram. methods. create\_invoice\_link. CreateInvoiceLink)                                      $			

```
attribute), 533
                                                                                                                                 attribute), 455
prices (aiogram.methods.send_invoice.SendInvoice at- protect_content(aiogram.methods.send_game.SendGame
                  tribute), 538
                                                                                                                                 attribute), 527
prices (aiogram.types.input_invoice_message_content.InputhoroicetMessageacontingram.methods.send_invoice.SendInvoice
                  attribute), 328
                                                                                                                                 attribute), 539
attribute), 343
                                                                                                                                 attribute), 458
PRIVATE (aiogram.enums.chat_type.ChatType attribute), protect_content (aiogram.methods.send_media_group.SendMediaGroup
                                                                                                                                 attribute), 460
prize_description(aiogram.types.giveaway.Giveaway protect_content(aiogram.methods.send_message.SendMessage
                  attribute), 140
                                                                                                                                 attribute), 463
prize_description (aiogram.types.giveaway_winners.GipanotayWinnantent (aiogram.methods.send_paid_media.SendPaidMedia
                  attribute), 143
                                                                                                                                 attribute), 465
prize_star_count (aiogram.types.chat_boost_source_givparotte_CluaBntontSoluniogFivecumathods.send_photo.SendPhoto
                                                                                                                                 attribute), 468
                  attribute), 57
prize_star_count (aiogram.types.giveaway.Giveaway protect_content(aiogram.methods.send_poll.SendPoll
                  attribute), 140
                                                                                                                                 attribute), 471
prize_star_count (aiogram.types.giveaway_created.GivepromyCatateontent (aiogram.methods.send_sticker.SendSticker
                                                                                                                                 attribute), 374
                  attribute), 141
prize_star_count (aiogram.types.giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway_winners.Giveaway
                  attribute), 142
                                                                                                                                 attribute), 474
profile_accent_color_id (aiogram.types.chat.Chat protect_content(aiogram.methods.send_video.SendVideo
                  attribute), 40
                                                                                                                                 attribute), 477
profile_accent_color_id
                                                                                                              protect_content(aiogram.methods.send video note.SendVideoNote
                  (aiogram.types.chat_full_info.ChatFullInfo
                                                                                                                                 attribute), 480
                  attribute), 59
                                                                                                              protect_content(aiogram.methods.send_voice.SendVoice
profile_background_custom_emoji_id
                                                                                                                                 attribute), 482
                  (aiogram.types.chat.Chat attribute), 40
                                                                                                              \verb"provider_data" (a iogram.methods.create_invoice\_link.CreateInvoiceLink) \\
profile_background_custom_emoji_id
                                                                                                                                 attribute), 534
                  (aiogram.types.chat_full_info.ChatFullInfo
                                                                                                              provider_data(aiogram.methods.send_invoice.SendInvoice
                  attribute), 59
                                                                                                                                 attribute), 538
promote() (aiogram.types.chat.Chat method), 47
                                                                                                              provider_data(aiogram.types.input_invoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoice_message_content.InputInvoic
PromoteChatMember
                                                                                                      in
                                                                                                                                 attribute), 328
                                                                                                              provider_payment_charge_id
                  aiogram.methods.promote_chat_member),
                                                                                                                                 (aiogram.types.refunded_payment.RefundedPayment
protect_content(aiogram.client.default.DefaultBotProperties
                                                                                                                                 attribute), 340
                  attribute), 567
                                                                                                              provider_payment_charge_id
protect_content(aiogram.methods.copy_message.CopyMessage
                                                                                                                                (aiogram.types.successful_payment.SuccessfulPayment
                  attribute), 393
                                                                                                                                 attribute), 346
protect_content (aiogram.methods.copy_messages.Copyphassiglesr_token (aiogram.methods.create_invoice_link.CreateInvoiceLink
                                                                                                                                 attribute), 533
                  attribute), 395
protect_content (aiogram.methods.forward_message.Fopproviidessageken (aiogram.methods.send_invoice.SendInvoice
                                                                                                                                 attribute), 538
                  attribute), 413
protect_content (aiogram.methods.forward_messages.Fopmowil@ess.ttpksen (aiogram.types.input_invoice_message_content.InputInv.
                                                                                                                                 attribute), 328
                  attribute), 414
protect_content (aiogram.methods.send_animation.Sendphroizintiotry_alert_radius
                                                                                                                                 (aiogram.methods.edit_message_live_location.EditMessageLiveL
                  attribute), 443
protect_content(aiogram.methods.send_audio.SendAudio
                                                                                                                                 attribute), 512
                  attribute), 446
                                                                                                              proximity_alert_radius
                                                                                                                                 (aiogram.methods.send_location.SendLocation
protect\_content (aiogram.methods.send\_contact.SendContact
                                                                                                                                 attribute), 457
                  attribute), 450
protect_content(aiogram.methods.send_dice.SendDice proximity_alert_radius
                  attribute), 452
                                                                                                                                 (aiogram.types.inline_query_result_location.InlineQueryResultLo
protect_content(aiogram.methods.send document.SendDocumentattribute), 315
```

proximity_alert_radius (aiogram.types.input_location_message_content. attribute), 330	quote_parse_mode(aiogram.types.reply_parameters.ReplyParameters InputLocatattt\(\mathbb{t}\) bearge \(\mathbb{C}\) intent quote_position(aiogram.types.reply_parameters.ReplyParameters
proximity_alert_radius	attribute), 272
(aiogram.types.location.Location attribute), 200	R
PROXIMITY_ALERT_TRIGGERED	react() (aiogram.types.message.Message method), 254
(aiogram.enums.content_type.ContentType attribute), 550	<pre>reaction(aiogram.methods.set_message_reaction.SetMessageReaction</pre>
<pre>proximity_alert_triggered</pre>	ReactionCount (class in
(aiogram.types.message.Message attribute),	aiogram.types.reaction_count), 268
209	${\tt reactions} \ (a iogram. types. message\_reaction\_count\_up dated. MessageReaction\_count\_up $
ProximityAlertTriggered (class in	attribute), 261
aiogram.types.proximity_alert_triggered), 267	ReactionType (class in aiogram.types.reaction_type), 268
PURCHASED_PAID_MEDIA	ReactionTypeCustomEmoji (class in
(aiogram.enums.update_type.UpdateType attribute), 562	aiogram.types.reaction_type_custom_emoji), 268
purchased_paid_media (aiogram.types.update.Update	ReactionTypeEmoji (class in
attribute), 362	aiogram.types.reaction_type_emoji), 269
PYG (aiogram.enums.currency.Currency attribute), 553	ReactionTypePaid (class in
Python Enhancement Proposals PEP 484,3	aiogram.types.reaction_type_paid), 269
PEP 404, 3 PEP 492, 3	ReactionTypeType (class in
FEF 492, 3	aiogram.enums.reaction_type_type), 560
Q	read() (aiogram.types.input_file.BufferedInputFile method), 185
QAR (aiogram.enums.currency.Currency attribute), 553	read() (aiogram.types.input_file.FSInputFile method),
query (aiogram.types.chosen_inline_result.ChosenInlineR	
attribute), 284	read() (aiogram.types.input_file.InputFile method), 184
query (aiogram.types.inline_query.InlineQuery at- tribute), 285	185
attribute), 274	w <b>itekbilineQuivgChastypEkst</b> ar_transaction.StarTransaction attribute), 345
query_id (aiogram.utils.web_app.WebAppInitData attribute), 655	tribute), 656
question (aiogram.methods.send_poll.SendPoll at- tribute), 470	RECORD_VIDEO (aiogram.enums.chat_action.ChatAction attribute), 547
question (aiogram.types.poll.Poll attribute), 265	${\tt record\_video()}\ (aiogram.utils.chat\_action.ChatActionSender$
${\tt question\_entities} \ (a iogram. methods. send\_poll. Send Polls \ and \ an$	
attribute), 470	RECORD_VIDEO_NOTE (aiogram.enums.chat_action.ChatAction
question_entities (aiogram.types.poll.Poll attribute),	attribute), 548
266	record_video_note()
question_parse_mode	(aiogram.utils.chat_action.ChatActionSender
(aiogram.methods.send_poll.SendPoll at- tribute), 470	class method), 652
QUIZ (aiogram.enums.keyboard_button_poll_type_type.Ke	RECORD_VOICE (aiogram.enums.chat_action.ChatAction
attribute), 556	record_voice() (aiogram.utils.chat_action.ChatActionSender
QUIZ (aiogram.enums.poll_type.PollType attribute), 559	class method), 652
quote (aiogram.types.message.Message attribute), 205	RED (aiogram.enums.topic_icon_color.TopicIconColor
quote (aiogram.types.reply_parameters.ReplyParameters	attribute), 561
attribute), 272	RedisStorage (class in aiogram.fsm.storage.redis), 600
${\tt quote\_entities} \ (a iogram. types. reply\_parameters. Reply Institute \ (a iogram. types. reply\_p$	PAREFUNDED_PAYMENT (aiogram.enums.content_type.ContentType
attribute) 272	attribute) 550

```
method), 213
refunded_payment (aiogram.types.message.Message
             attribute), 208
                                                                                  reply_contact() (aiogram.types.inaccessible message.InaccessibleMess
RefundedPayment
                                                (class
                                                                            in
                                                                                                method), 151
             aiogram.types.refunded_payment), 340
                                                                                  reply_contact()
                                                                                                                   (aiogram.types.message.Message
RefundStarPayment
                                                 (class
                                                                            in
                                                                                                method), 216
             aiogram.methods.refund star payment),
                                                                                  reply_dice() (aiogram.types.inaccessible message.InaccessibleMessage
                                                                                                method), 169
regexp_match(aiogram.filters.command.CommandObjectreply_dice()
                                                                                                                    (aiogram.types.message.Message
              attribute), 580
                                                                                                method), 235
register() (aiogram.fsm.scene.SceneRegistry method), reply_document() (aiogram.types.inaccessible_message.InaccessibleMessage.
                                                                                                method), 153
register() (aiogram.webhook.aiohttp_server.BaseRequestdeptdedocument()
                                                                                                                   (aiogram.types.message.Message
             method), 630
                                                                                                method), 217
register() (aiogram.webhook.aiohttp_server.SimpleReqursathUndtyame() (aiogram.types.inaccessible_message.InaccessibleMessage
                                                                                                method), 155
             method), 630
register() (aiogram.webhook.aiohttp_server.TokenBasedReplesstHamel(e)r
                                                                                                                    (aiogram.types.message.Message
                                                                                                method), 220
             method), 631
REGULAR (aiogram.enums.keyboard_button_poll_type_type.KephsyaridBudioaFQ)[(hipeFupe.types.inaccessible_message.InaccessibleMess
                                                                                                method), 157
             attribute), 556
REGULAR (aiogram.enums.poll type.PollType attribute), reply_invoice()
                                                                                                                   (aiogram.types.message.Message
                                                                                                method), 221
REGULAR
                      (aiogram.enums.sticker_type.StickerType reply_location() (aiogram.types.inaccessible_message.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleM
                                                                                                method), 160
             attribute), 561
remove_caption(aiogram.methods.copy messages.CopyMacrolyeslocation()
                                                                                                                   (aiogram.types.message.Message
             attribute), 395
                                                                                                method), 225
remove_date (aiogram.types.chat_boost_removed.ChatBoosepdnyomarkup (aiogram.methods.copy_message.CopyMessage
             attribute), 55
                                                                                                attribute), 393
remove_keyboard (aiogram.types.reply_keyboard_remove_keplyKaydxkaydRaingvam.methods.edit_message_caption.EditMessageCapt
             attribute), 271
                                                                                                attribute), 510
REMOVED_CHAT_BOOST (aiogram.enums.update_type.UpdataEppley_markup (aiogram.methods.edit_message_live_location.EditMessage
             attribute), 562
                                                                                                attribute), 512
removed_chat_boost(aiogram.types.update.Update at- reply_markup(aiogram.methods.edit_message_media.EditMessageMedia
              tribute), 362
                                                                                                attribute), 514
                                                                                  reply_markup(aiogram.methods.edit_message_reply_markup.EditMessage
render() (aiogram.utils.formatting.Text method), 665
RENTAL_AGREEMENT (aiogram.enums.encrypted_passport_
                                                                                 element.EnattviltatkPasssportElement
             attribute), 555
                                                                                  reply_markup(aiogram.methods.edit_message_text.EditMessageText
ReopenForumTopic
                                                (class
                                                                            in
                                                                                                attribute), 517
             aiogram.methods.reopen_forum_topic), 437
                                                                                  reply_markup (aiogram.methods.send_animation.SendAnimation
ReopenGeneralForumTopic
                                                       (class
                                                                            in
                                                                                                attribute), 443
             aiogram.methods.reopen_general_forum_topic), reply_markup(aiogram.methods.send_audio.SendAudio
             438
                                                                                                attribute), 446
ReplaceStickerInSet
                                                   (class
                                                                                  reply_markup(aiogram.methods.send contact.SendContact
             aiogram.methods.replace_sticker_in_set),
                                                                                                attribute), 450
                                                                                  \verb"reply_markup" (a iogram.methods.send_dice.SendDice")
reply() (aiogram.types.inaccessible_message.InaccessibleMessage attribute), 452
              method), 144
                                                                                  reply_markup(aiogram.methods.send_document.SendDocument
reply() (aiogram.types.message.Message method), 228
                                                                                                attribute), 455
reply_animation() (aiogram.types.inaccessible_messagerleplye.methods.send_game.SendGame
             method), 146
                                                                                                attribute), 527
reply_animation() (aiogram.types.message.Message reply_markup(aiogram.methods.send_invoice.SendInvoice
                                                                                                attribute), 539
             method), 211
reply_audio() (aiogram.types.inaccessible_message.Inacresslbletheskupe(aiogram.methods.send_location.SendLocation
             method), 149
                                                                                                attribute), 458
reply_audio()
                                 (aiogram.types.message.Message reply_markup(aiogram.methods.send message.SendMessage
```

```
attribute), 463
                                                                                                  attribute), 319
reply_markup(aiogram.methods.send_paid_media.SendPaidMedimarkup(aiogram.types.inline_query_result_venue.InlineQueryResu
             attribute), 465
                                                                                                  attribute), 321
reply_markup(aiogram.methods.send_photo.SendPhoto reply_markup(aiogram.types.inline_query_result_video.InlineQueryResu
             attribute), 468
                                                                                                  attribute), 323
reply_markup(aiogram.methods.send_poll.SendPoll at- reply_markup(aiogram.types.inline_query_result_voice.InlineQueryResu
                                                                                                  attribute), 325
             tribute), 471
reply_markup (aiogram.methods.send_sticker.SendSticker reply_markup (aiogram.types.message.Message at-
             attribute), 374
                                                                                                  tribute), 210
reply_markup(aiogram.methods.send_venue.SendVenue reply_media_group()
             attribute), 474
                                                                                                  (aiogram.types.inaccessible_message.InaccessibleMessage
reply_markup(aiogram.methods.send_video.SendVideo
                                                                                                  method), 162
             attribute), 477
                                                                                   reply_media_group()
reply_markup(aiogram.methods.send_video_note.SendVideoNote
                                                                                                 (aiogram.types.message.Message
                                                                                                                                                        method),
                                                                                                  227
             attribute), 480
reply_markup (aiogram.methods.send_voice.SendVoice reply_paid_media() (aiogram.types.inaccessible_message.InaccessibleN
             attribute), 482
                                                                                                  method), 182
reply_markup(aiogram.methods.stop_message_live_locatioepplreppleisdamedivek)c(atiogram.types.message.Message
             attribute), 519
                                                                                                  method), 255
reply_markup (aiogram.methods.stop_poll.StopPoll at- reply_parameters (aiogram.methods.copy_message.CopyMessage
             tribute), 520
                                                                                                  attribute), 393
reply_markup (aiogram.types.inline_query_result_article.lndpleQpay&naxleAss(cdicogram.methods.send_animation.SendAnimation
                                                                                                  attribute), 443
              attribute), 287
reply_markup(aiogram.types.inline_query_result_audio.InlinedQupparlementsexst(aiogram.methods.send_audio.SendAudio
             attribute), 290
                                                                                                  attribute), 446
reply_markup(aiogram.types.inline_query_result_cached_nepliy_Iphina@etexRdxult@muchad&thdits.send_contact.SendContact
             attribute), 292
                                                                                                  attribute), 450
reply_markup(aiogram.types.inline_query_result_cached_ndmplynaputdekiet@csetylResultaCondthdDosemule_ndice.SendDice
             attribute), 294
                                                                                                  attribute), 452
reply_markup(aiogram.types.inline_query_result_cached_peif_lylipa@aweylexsakt@ioghadCrifethods.send_document.SendDocument
             attribute), 296
                                                                                                  attribute), 455
reply_markup (aiogram.types.inline_query_result_cached_reply_4 peif.hanlere@sdxyiResultnCinethadlM.pegd_Cinfime.SendGame
             attribute), 298
                                                                                                  attribute), 527
reply_markup(aiogram.types.inline_query_result_cached_pdpky_Ipkine@erexRe(ailt@wwhedPhotds.send_invoice.SendInvoice
             attribute), 300
                                                                                                  attribute), 539
reply_markup(aiogram.types.inline_query_result_cached_reightyr_plinenQtenyResidg@awchand&hiodlessend_location.SendLocation
             attribute), 302
                                                                                                  attribute), 458
reply_markup (aiogram.types.inline_query_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cac
              attribute), 304
                                                                                                  attribute), 460
reply_markup(aiogram.types.inline_query_result_cached_nepides_Iplina@ete_nResult@cached_nebidioels.send_message.SendMessage
                                                                                                  attribute), 463
             attribute), 306
reply_markup(aiogram.types.inline_query_result_contact:#apikyQparyMestdtSchutagram.methods.send_paid_media.SendPaidMedia
             attribute), 308
                                                                                                  attribute), 465
reply_markup (aiogram.types.inline_query_result_documenephlynp@nemyttesnstiQnonymen.tmethods.send_photo.SendPhoto
             attribute), 310
                                                                                                  attribute), 468
reply_markup(aiogram.types.inline_query_result_game.Indiape\DuparRamah\Grandaiogram.methods.send_poll.SendPoll
             attribute), 311
                                                                                                  attribute), 471
reply_markup(aiogram.types.inline_query_result_gif.Inline@pley.yRaraln@tfers(aiogram.methods.send_sticker.SendSticker
             attribute), 313
                                                                                                  attribute), 374
reply_markup(aiogram.types.inline_query_result_location:bpline@parankerealts.cainigram.methods.send_venue.SendVenue
             attribute), 315
                                                                                                  attribute), 474
reply_markup(aiogram.types.inline_query_result_mpeg4_replyinpa@nemyRersultMpgg466ifnethods.send_video.SendVideo
             attribute), 317
                                                                                                  attribute), 477
reply_markup(aiogram.types.inline query result photo.IndepelQupearRanatiethes(aiogram.methods.send video note.SendVideoNote
```

attribute), 480		(aiogram.methods.send_po	oll.SendPoll	at-
reply_parameters (aiogram.methods.send_voice.SendVo		tribute), 472		
attribute), 482		o_message_id		
reply_photo() (aiogram.types.inaccessible_message.Inac method), 164	ccessibleN	A <b>exsage</b> am.methods.send_st attribute), 374	icker.SendSticker	
reply_photo() (aiogram.types.message.Message	renly t			
method), 230	rcpry_c	(aiogram.methods.send_ve	nue SendVenue	
reply_poll() (aiogram.types.inaccessible_message.Inacc	ressibleM		пис. Зепа чепие	
method), 167		o_message_id		
reply_poll() (aiogram.types.message.Message	rcpry_c	(aiogram.methods.send_vi	den SendViden	
method), 232		attribute), 477	ueo.sena viaeo	
reply_sticker() (aiogram.types.inaccessible_message.L	mondarit			
method), 171	пи <b>ер</b> азу <u>п</u> и	_	daa nota SandVid	aoNota
		(aiogram.methods.send_vi	aeo_noie.senavia	eonoie
reply_sticker() (aiogram.types.message.Message		attribute), 480		
method), 236	repry_t	o_message_id	· C W:	
reply_to_message (aiogram.types.message.Message		(aiogram.methods.send_vo	nce.SendVoice	
attribute), 205	_	attribute), 483		
reply_to_message_id	reply_t	o_story (aiogram.types.n	nessage.Message	at-
(aiogram.methods.copy_message.CopyMessage		tribute), 206		
attribute), 393	reply_v	enue() (aiogram.types.ina	ccessible_message	e.InaccessibleMessa
reply_to_message_id		method), 173		
$(a iogram. methods. send\_animation. Send Animation) \\$	<i>n</i> reply_v	enue() (aiogram.typ	oes.message.Messo	age
attribute), 444		<i>method</i> ), 238		
reply_to_message_id	reply_v	ideo() (aiogram.types.ina	ccessible_message	.InaccessibleMessa
(aiogram.methods.send_audio.SendAudio		method), 175		
attribute), 446	reply_v	ideo() (aiogram.ty)	oes.message.Messa	age
reply_to_message_id		method), 239		
(aiogram.methods.send_contact.SendContact	reply_v	ideo_note()(aiogram.typ	es.inaccessible m	essage.Inaccessible
attribute), 450		method), 177		Ü
reply_to_message_id	reply_v	ideo_note() (aiogram.typ	oes.message.Messa	age
(aiogram.methods.send_dice.SendDice at-	. , –	method), 242	O O	O
tribute), 452	replv v	oice() (aiogram.types.ina	ccessible message	.InaccessibleMessa
reply_to_message_id	1 7 –	method), 180		
(aiogram.methods.send_document.SendDocumen	treply v		oes.message.Messa	196
attribute), 455	ni cpij_•	method), 244	esimessuze.171esse	.80
reply_to_message_id	RenlyKe	yboardBuilder	(class	in
(aiogram.methods.send_game.SendGame	пертупе	aiogram.utils.keyboard), 6		
attribute), 527	DanlyKa	yboardMarkup	(class	in
reply_to_message_id	кертуке	aiogram.types.reply_keybo	`	iii
(aiogram.methods.send_invoice.SendInvoice		270	<i>α</i> ια_ <i>т</i> αικиρ),	
	Dan lu.Va		(alass	:
attribute), 539	кертуке	yboardRemove	(class	in
reply_to_message_id	n 1 n	aiogram.types.reply_keybo		
(aiogram.methods.send_location.SendLocation	кертуРа	*	lass	in
attribute), 458		aiogram.types.reply_paran		ID
reply_to_message_id		_chat (aiogram.types.keyb	oard_button.Keyb	oardButton
(aiogram.methods.send_media_group.SendMedia		attribute), 194		
attribute), 460	request	_contact(aiogram.types.k	xeyboard_button.K	eyboardButton
reply_to_message_id		attribute), 194		
(aiogram.methods.send_message.SendMessage	request	_id (aiogram.types.chat_sh	ared.ChatShared	at-
attribute), 463		tribute), 132		
reply_to_message_id	request	_id(aiogram.types.keyboar	rd_button_request	_chat.KeyboardBut
$(aiogram.methods.send\_photo.SendPhoto$		attribute), 196		
attribute), 468	request	_id(aiogram.types.keyboar	rd_button_request	_user.KeyboardBut
reply_to_message_id		attribute), 197		

```
request_id(aiogram.types.keyboard_button_request_users.KeyboardBritattRequestUsers
              attribute), 198
                                                                                     result (aiogram.methods.answer_web_app_query.AnswerWebAppQuery
request_id (aiogram.types.user shared.UserShared at-
                                                                                                    attribute), 524
                                                                                     {\tt result\_id}(aiogram.types.chosen\_inline\_result.ChosenInlineResult
              tribute), 277
request_id (aiogram.types.users_shared.UsersShared
                                                                                                    attribute), 284
              attribute), 278
                                                                                     results (aiogram.methods.answer inline query.AnswerInlineQuery
request_location(aiogram.types.keyboard button.KeyboardButtomttribute), 522
                                                                                     retake() (aiogram.fsm.scene.SceneWizard method), 620
              attribute), 194
request_name (aiogram.types.keyboard_button_request_usext.KeybotaetRuitogrkeustysetsl.sesponse_parameters.ResponseParameters
                                                                                                    attribute), 272
              attribute), 198
request_photo(aiogram.types.keyboard_button_request_klevekkeybiinthlBnewalk&queseChat
                                                                                                                                              (class
                                                                                                                                                                     in
                                                                                                    aiogram.types.revenue_withdrawal_state),
              attribute), 196
request_photo(aiogram.types.keyboard_button_request_users.KeyboardButtonRequestUsers
              attribute), 198
                                                                                     RevenueWithdrawalStateFailed
                                                                                                                                                   (class
                                                                                                                                                                     in
request\_poll\ (aiogram.types.keyboard\_button.KeyboardButton
                                                                                                    aiogram.types.revenue_withdrawal_state_failed),
              attribute), 194
                                                                                                    341
request_title (aiogram.types.keyboard_button_request_Revellentling Revellentling
                                                                                                                                                                     in
                                                                                                                                                    (class
              attribute), 196
                                                                                                    aiogram.types.revenue_withdrawal_state_pending),
request\_user (aiogram.types.keyboard\_button.KeyboardButton
                                                                                     RevenueWithdrawalStateSucceeded
              attribute), 194
                                                                                                                                                      (class
                                                                                                                                                                     in
request_username(aiogram.types.keyboard_button_request_chat.KaibbynamlBuptcmReagnext_Ghithdrawal_state_succeeded),
              attribute), 196
request_username(aiogram.types.keyboard_button_request_wentedbutstoneredistantedbutstoneredistantedbutstoneredistantedbutstoneredistantedbutstoneredistantedbutstoneredistantedbutstoneredistantedbutstoneredistantedbutstoneredistantedbutstoneredistantedbutstoneredistantedbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutstoneredbutst
                                                                                                                                                                     in
              attribute), 198
                                                                                                    aiogram.enums.revenue withdrawal state type),
request_users (aiogram.types.keyboard_button.KeyboardButton
              attribute), 194
                                                                                     REVERSE_SIDE (aiogram.enums.passport_element_error_type.PassportEle.
request_write_access
                                                                                                    attribute), 559
              (aiogram.types.login_url.LoginUrl
                                                                   attribute),
                                                                                     reverse_side(aiogram.types.encrypted_passport_element.EncryptedPass
                                                                                                    attribute), 351
                                                                                     revoke_invite_link()
reset_data_on_enter
                                                                                                                                   (aiogram.types.chat.Chat
              (aiogram.fsm.scene.SceneConfig
                                                                    attribute),
                                                                                                    method), 42
              619
                                                                                     {\tt revoke\_messages} \ (a iogram. methods. ban\_chat\_member. BanChatMember
                                                                                                    attribute), 387
reset_history_on_enter
                                                                                     RevokeChatInviteLink
              (aiogram.fsm.scene.SceneConfig
                                                                   attribute),
                                                                                                                                            (class
                                                                                                                                                                     in
                                                                                                    aiogram.methods.revoke_chat_invite_link),
resize_keyboard(aiogram.types.reply_keyboard_markup.ReplyKeyboardMarkup)
              attribute), 270
                                                                                     rights (aiogram.methods.set my default administrator rights.SetMyDefa
resolve_bot() (aiogram.webhook.aiohttp_server.BaseRequestHandbatribute), 495
              method), 630
                                                                                     RON (aiogram.enums.currency.Currency attribute), 553
{\tt resolve\_bot()}\ (aiogram.webhook.aiohttp\_server.Simple \textit{RRQSE}st \textit{Haia} \textit{Hearm.enums.topic\_icon\_color.TopicIconColor}
              method), 631
                                                                                                    attribute), 561
resolve_bot() (aiogram.webhook.aiohttp_server.TokenBassotlRziques_blasgllee(aiogram.types.background_fill_gradient.BackgroundFill
              method), 631
                                                                                                    attribute), 22
resolve_used_update_types()
                                                                                     Router (class in aiogram.dispatcher.router), 569
              (aiogram.dispatcher.router.Router
                                                                                                   (aiogram.utils.keyboard.InlineKeyboardBuilder
                                                                     method),
                                                                                     row()
              570
                                                                                                    method), 645
                                                                                                    (aiogram.utils.keyboard.ReplyKeyboardBuilder
ResponseParameters
                                                    (class
                                                                               in row()
              aiogram.types.response_parameters), 272
                                                                                                    method), 647
RestartingTelegram, 626
                                                                                     RSD (aiogram.enums.currency.Currency attribute), 553
restrict() (aiogram.types.chat.Chat method), 48
                                                                                     RUB (aiogram.enums.currency.Currency attribute), 553
RestrictChatMember
                                                                                     run_polling() (aiogram.dispatcher.dispatcher.Dispatcher
                                                    (class
              aiogram.methods.restrict_chat_member),
                                                                                                   method), 576
```

RESTRICTED (aiogram.enums.chat member status.ChatMemberStatus

S	send_phone_number_to_provider
<pre>safe_parse_webapp_init_data() (in module</pre>	(aiogram.types.input_invoice_message_content.InputInvoiceMess
aiogram.utils.web_app), 654	attribute), 329
SAR (aiogram.enums.currency.Currency attribute), 553	SendAnimation (class in
scale (aiogram.types.mask_position.MaskPosition at-	aiogram.methods.send_animation), 442
tribute), 334	SendAudio (class in aiogram.methods.send_audio), 445
Scene (class in aiogram.fsm.scene), 617	SendChatAction (class in
SceneConfig (class in aiogram.fsm.scene), 619	aiogram.methods.send_chat_action), 447
SceneException, 626	SendContact (class in aiogram.methods.send_contact),
SceneRegistry (class in aiogram.fsm.scene), 617	449
ScenesManager (class in aiogram.fsm.scene), 618	SendDice (class in aiogram.methods.send_dice), 451
SceneWizard (class in aiogram.fsm.scene), 619	SendDocument (class in
scope (aiogram.methods.delete_my_commands.DeleteMyC	Commands aiogram.methods.send_document), 453
attribute), 405	SENDER (aiogram.enums.chat_type.ChatType attribute),
scope (aiogram.methods.get_my_commands.GetMyComma	5.40
attribute), 424	sender_boost_count (aiogram.types.message.Message
scope (aiogram.methods.set_my_commands.SetMyComma	4. 4. 1 205
	sender_business_bot
attribute), 494	(aiogram.types.message.Message attribute),
score (aiogram.methods.set_game_score.SetGameScore	205
attribute), 529	sender_chat (aiogram.types.message.Message at-
score (aiogram.types.game_high_score.GameHighScore	tribute), 205
attribute), 365	sender chat (aiogram.types.message_origin_chat.MessageOriginChat
Secret (alogram.types.encryptea_creaentials.EncrypteaCi	attribute), 260
attribute), 350	ksender_chat_id (aiogram.methods.ban_chat_sender_chat.BanChatSende
secret_token(alogram.methods.set_webhook.SetWebhook	attribute), 388
attribute), 544	sender_chat_id (aiogram.methods.unban_chat_sender_chat.UnbanChat
SEK (aiogram.enums.currency.Currency attribute), 553	attribute), 501
selective (aiogram.types.force_reply.ForceReply	sender_user(aiogram.types.message_origin_user.MessageOriginUser
attribute), 137	
${\tt selective} (a iogram. types. reply\_keyboard\_markup. ReplyReplyReplyReplyReplyReplyReplyReply$	KeyboardMdtk\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
attribute), 270	
${\tt selective} ({\it aiogram.types.reply\_keyboard\_remove.ReplyKeyboard\_re$	(eyboardRetttoVeure), 200
attribute), 271	SendGame (class in aiogram.methods.send_game), 526
SELFIE (aiogram.enums.passport_element_error_type.Pas.	spontetementeriorassen alogram.memoas.sena_invoice),
attribute), 559	337
${\tt selfie} (a iogram. types. encrypted\_passport\_element. Encry$	psendiocation (class in
attribute), 351	aiogram.methods.send_location), 456
send_copy() (aiogram.types.message.Message method),	SendMediaGroup (class in
246	aiogram.methods.send_media_group), 459
send_email_to_provider	SendMessage (class in aiogram.methods.send_message),
(aiogram.methods.create_invoice_link.CreateInvoice_	piceLink 461
attribute), 534	SendPaidMedia (class in
send_email_to_provider	aiogram.methods.send_paid_media), 464
(aiogram.methods.send_invoice.SendInvoice	SendPhoto (class in aiogram.methods.send_photo), 466
attribute), 539	SendPol1 (class in aiogram.methods.send_poll), 469
send_email_to_provider	SendSticker (class in aiogram.methods.send_sticker),
(aiogram.types.input_invoice_message_content.In	nputInvoiceMessageContent SendVenue (class in aiogram.methods.send_venue), 473
attribute), 329	
send_phone_number_to_provider	SendVideo (class in aiogram.methods.send_video), 475
(aiogram.methods.create_invoice_link.CreateInvo	SendVideoNote (class in
attribute), 534	aiogram.metnoas.sena_viaeo_noie), 479
send_phone_number_to_provider	SendVoice (class in aiogram.methods.send_voice), 481
(aiogram.methods.send_invoice.SendInvoice	SentWebAppMessage (class in
attribute). 539	aiogram.types.sent_web_app_message), 333

<pre>set_administrator_custom_title()</pre>	497
(aiogram.types.chat.Chat method), 46	SetMyShortDescription (class in
<pre>set_data() (aiogram.fsm.scene.SceneWizard method), 620</pre>	aiogram.methods.set_my_short_description), 498
set_data() (aiogram.fsm.storage.base.BaseStorage	10.00
method), 602	aiogram.methods.set_passport_data_errors),
$\verb set_description()  (a iogram. types. chat. Chat method), \\$	545
50	SetStickerEmojiList (class in
set_locale() (aiogram.utils.i18n.middleware.FSMI18n.method), 649	Middleware aiogram.methods.set_sticker_emoji_list), 376
<pre>set_name (aiogram.types.sticker.Sticker attribute), 335</pre>	SetStickerKeywords (class in
<pre>set_permissions() (aiogram.types.chat.Chat method), 47</pre>	aiogram.methods.set_sticker_keywords), 377
<pre>set_photo() (aiogram.types.chat.Chat method), 50</pre>	SetStickerMaskPosition (class in
<pre>set_position_in_set() (aiogram.types.sticker.Sticker method), 335</pre>	aiogram.methods.set_sticker_mask_position), 378
<pre>set_state() (aiogram.fsm.storage.base.BaseStorage</pre>	SetStickerPositionInSet (class in
method), 602	aiogram.methods.set_sticker_position_in_set),
<pre>set_sticker_set() (aiogram.types.chat.Chat method),</pre>	379
44	SetStickerSetThumbnail (class in
<pre>set_title() (aiogram.types.chat.Chat method), 50 SetChatAdministratorCustomTitle (class in</pre>	aiogram.methods.set_sticker_set_thumbnail), 380
aiogram.methods.set_chat_administrator_custor	
484	aiogram.methods.set_sticker_set_title), 382
	setup() (aiogram.utils.i18n.middleware.118nMiddleware
aiogram.methods.set_chat_description), 485	method), 650
SetChatMenuButton (class in	SetWebhook (class in aiogram.methods.set_webhook),
aiogram.methods.set_chat_menu_button),	543
486	SGD (aiogram.enums.currency.Currency attribute), 553
SetChatPermissions (class in	
aiogram.methods.set_chat_permissions),	shifted_id (aiogram.types.chat.Chat property), 40
487	shipping_address(aiogram.types.order_info.OrderInfo
SetChatPhoto (class in	attribute), 338
aiogram.methods.set_chat_photo), 489	$\verb shipping_address   (a iogram. types. shipping_query. ShippingQuery$
SetChatStickerSet (class in	attribute), 344
aiogram.methods.set_chat_sticker_set), 490	shipping_option_id(aiogram.types.pre_checkout_query.PreCheckouty
SetChatTitle (class in	attribute), 339
aiogram.methods.set_chat_title), 491	shipping_option_id(aiogram.types.successful_payment.SuccessfulPay
SetCustomEmojiStickerSetThumbnail (class in	attribute), 346
aiogram.methods.set_custom_emoji_sticker_set_ 375	_thahilppaing_options (aiogram.methods.answer_shipping_query.AnswerSh attribute), 532
SetGameScore (class in	SHIPPING_QUERY (aiogram.enums.update_type.UpdateType
aiogram.methods.set_game_score), 529	attribute), 562
SetMessageReaction (class in	shipping_query (aiogram.types.update.Update at-
aiogram.methods.set_message_reaction),	tribute), 362
492	shipping_query_id(aiogram.methods.answer_shipping_query.Answer_
SetMyCommands (class in	attribute), 531
aiogram.methods.set_my_commands), 493	ShippingAddress (class in
SetMyDefaultAdministratorRights (class in	aiogram.types.shipping_address), 343
aiogram.methods.set_my_default_administrator_	
495 SetMyDescription (class in	aiogram.types.shipping_option), 343 ShippingQuery (class in
aiogram.methods.set_my_description), 496	aiogram.types.shipping_query), 344
SetMyName (class in aiogram.methods.set_my_name),	
, (cross on arcgrammemous.set_my_name),	

 ${\tt source} \ (aiogram.types.passport\_element\_error\_translation\_files. PassportIndex and the property of the$ 

attribute), 498	show_caption_above_media
short_description(aiogram.types.bot_short_description attribute), 31	on.BotShort <b>Quiscripticty</b> pes.input_media_animation.InputMediaAnimation attribute), 187
show_above_text (aiogram.types.link_preview_options.l	
attribute), 199	(aiogram.types.input_media_photo.InputMediaPhoto
show_alert(aiogram.methods.answer_callback_query.A.	
attribute), 384	show_caption_above_media
show_alert(aiogram.utils.callback_answer.CallbackAns	
property), 661	attribute), 191
show_caption_above_media	show_caption_above_media
(aiogram.client.default.DefaultBotProperties attribute), 567	(aiogram.types.message.Message attribute), 207
show_caption_above_media	SimpleI18nMiddleware (class in
(aiogram.methods.copy_message.CopyMessage	aiogram.utils.i18n.middleware), 648
attribute), 393	SimpleRequestHandler (class in
show_caption_above_media	aiogram.webhook.aiohttp_server), 630
(aiogram.methods.edit_message_caption.EditMe	es <b>SigeT_ApaGH</b> iINE (aiogram.enums.dice_emoji.DiceEmoji
attribute), 510	attribute), 554
show_caption_above_media	SLOT_MACHINE (aiogram.types.dice.DiceEmoji at-
(aiogram.methods.send_animation.SendAnimati	
attribute), 443	slow_mode_delay (aiogram.types.chat.Chat attribute),
show_caption_above_media	40
	edallow_mode_delay(aiogram.types.chat_full_info.ChatFullInfo
attribute), 465	attribute), 60
show_caption_above_media	small_file_id (aiogram.types.chat_photo.ChatPhoto
(aiogram.methods.send_photo.SendPhoto	attribute), 131
<pre>attribute), 467 show_caption_above_media</pre>	small_file_unique_id
(aiogram.methods.send_video.SendVideo	(aiogram.types.chat_photo.ChatPhoto at- tribute), 131
attribute), 477	source (aiogram.types.chat_boost.ChatBoost attribute),
show_caption_above_media	54
	InBoxQve(vResulti).tighealChift_boost_removed.ChatBoostRemoved
attribute), 296	attribute), 55
show_caption_above_media	source (aiogram.types.chat_boost_source_gift_code.ChatBoostSourceGif
(aiogram.types.inline_query_result_cached_mpe	
attribute), 298	source (aiogram.types.chat_boost_source_giveaway.ChatBoostSourceGiv
show_caption_above_media	attribute), 56 to <b>sbubineQueingRamults@ashddtl\plot</b> oost_source_premium.ChatBoostSourcePren
attribute), 300	io <b>ssimuse-gataogsiasiaispastaaii <u>r</u>uoio</b> usi_source_premium.Cnaiboosisourcei rei attribute), 57
show_caption_above_media	source (aiogram.types.passport_element_error_data_field.PassportEleme
(aiogram.types.inline_query_result_cached_vide	
attribute), 304	source(aiogram.types.passport_element_error_file.PassportElementErro
show_caption_above_media	attribute), 353
	ter <b>sRusalt(aj</b> fogram.types.passport_element_error_files.PassportElementErro
attribute), 313	attribute), 354
show_caption_above_media	source (aiogram.types.passport_element_error_front_side.PassportEleme
(aiogram.types.inline_query_result_mpeg4_gif.1	
attribute), 317	source (aiogram.types.passport_element_error_reverse_side.PassportElement_error_revers
show_caption_above_media	attribute), 356
	QuouxRes(diPlyatm.types.passport_element_error_selfie.PassportElementEr.
attribute), 319	attribute), 357
show_caption_above_media	source(aiogram.types.passport_element_error_translation_file.Passportl
(aiooram tynes inline auery result video Inline	

Index 795

attribute), 323

```
attribute), 358
                                                                                                attribute), 372
{\tt source} \ (aiogram.types.passport\_element\_error\_unspecified \verb|SPaissport|EdwingnamEmmethbulspseifit| \underline{c} sticker. Send Sticker\ attacker. Send S
             attribute), 359
                                                                                                tribute), 373
source (aiogram.types.star_transaction.StarTransaction sticker(aiogram.methods.set_sticker_emoji_list.SetStickerEmojiList
              attribute), 345
                                                                                                 attribute), 376
SPOILER (aiogram.enums.message_entity_type.MessageEntistTydeer (aiogram.methods.set_sticker_keywords.SetStickerKeywords
             attribute), 558
                                                                                                attribute), 377
Spoiler (class in aiogram.utils.formatting), 667
                                                                                   sticker(aiogram.methods.set_sticker_mask_position.SetStickerMaskPosit
star_count(aiogram.methods.send_paid_media.SendPaidMedia
                                                                                                attribute), 378
                                                                                   attribute), 464
{\tt star\_count}\ (aiogram.types.paid\_media\_info.PaidMediaInfo
                                                                                                attribute), 379
             attribute), 263
                                                                                   sticker(aiogram.methods.upload_sticker_file.UploadStickerFile
start_date(aiogram.types.video_chat_scheduled.VideoChatScheduletdribute), 383
             attribute), 281
                                                                                   sticker (aiogram.types.business_intro.BusinessIntro at-
start_param (aiogram.utils.web_app.WebAppInitData
                                                                                                tribute), 32
                                                                                   \verb+sticker+ (a iogram.types.external\_reply\_info. ExternalReplyInfo
             attribute), 656
start_parameter(aiogram.methods.send_invoice.SendInvoice
                                                                                                attribute), 135
             attribute), 538
                                                                                   sticker
                                                                                                         (aiogram.types.input_sticker.InputSticker
start_parameter(aiogram.types.inline_query_results_button.Inline@trebuRe)sidlsSButton
             attribute), 325
                                                                                   sticker (aiogram.types.message.Message attribute), 206
start_parameter
                                    (aiogram.types.invoice.Invoice
                                                                                  Sticker (class in aiogram.types.sticker), 334
             attribute), 337
                                                                                   sticker_file_id(aiogram.types.inline_query_result_cached_sticker.Inli
start_polling() (aiogram.dispatcher.dispatcher.Dispatcher
                                                                                                attribute), 302
             method), 576
                                                                                   sticker_format(aiogram.methods.create new sticker set.CreateNewSticker)
StarTransaction
                                                                                                attribute), 367
                                                (class
                                                                            in
             aiogram.types.star_transaction), 345
                                                                                  sticker_format(aiogram.methods.upload_sticker_file.UploadStickerFile
StarTransactions
                                                (class
                                                                            in
                                                                                                attribute), 383
             aiogram.types.star_transactions), 345
                                                                                  state (aiogram.fsm.scene.SceneConfig attribute), 619
                                                                                                attribute), 490
state(aiogram.types.shipping_address.ShippingAddress
                                                                                  sticker_set_name (aiogram.types.chat.Chat attribute),
             attribute), 343
STATIC
               (aiogram.enums.sticker_format.StickerFormat sticker_set_name(aiogram.types.chat_full_info.ChatFullInfo
             attribute), 560
                                                                                                attribute), 61
status (aiogram.types.chat_member_administrator.ChatMestibek*administratorgram.methods.create_new_sticker_set.CreateNewSticke
             attribute), 104
                                                                                                 attribute), 367
status (aiogram.types.chat_member_banned.ChatMemberBtinker_type (aiogram.types.sticker_set.StickerSet at-
             attribute), 106
                                                                                                tribute), 336
status(aiogram.types.chat_member_left.ChatMemberLeftStickerFormat
                                                                                                                                                               in
                                                                                                                                 (class
              attribute), 106
                                                                                                aiogram.enums.sticker_format), 560
status (aiogram.types.chat_member_member.ChatMembersMainkers (aiogram.methods.create_new_sticker_set.CreateNewStickerSet
                                                                                                attribute), 367
             attribute), 107
status (aiogram.types.chat_member_owner.ChatMemberOsxtieckers
                                                                                                      (aiogram.types.sticker_set.StickerSet
                                                                                                                                                              at-
                                                                                                tribute), 336
             attribute), 107
status (aiogram.types.chat_member_restricted.ChatMemb&rtlieskeits&elt (class in aiogram.types.sticker_set), 336
                                                                                   StickerType (class in aiogram.enums.sticker_type), 561
             attribute), 108
STICKER (aiogram.enums.content_type.ContentType at- stop_live_location()
             tribute), 549
                                                                                                 (aiogram.types.message.Message
                                                                                                                                                      method),
STICKER (aiogram.enums.inline_query_result_type.InlineQueryResult_type
             attribute), 555
                                                                                   stop_polling() (aiogram.dispatcher.dispatcher.Dispatcher
sticker(aiogram.methods.add\_sticker\_to\_set.AddStickerToSet
                                                                                                method), 576
                                                                                   StopMessageLiveLocation
                                                                                                                                          (class
                                                                                                                                                               in
             attribute), 365
sticker (aiogram.methods.delete_sticker_from_set.DeleteStickerFromSegram.methods.stop_message_live_location),
             attribute), 368
sticker (aiogram.methods.replace sticker in set.Replace\State\Phi\Delta\text{tdpPhi\Delta} (class in aiogram.methods.stop poll), 520
```

```
STORY (aiogram.enums.content_type.ContentType at- supports_streaming(aiogram.types.input_media_video.InputMediaVide
              tribute), 549
                                                                                                   attribute), 191
story (aiogram.types.external reply info.ExternalReplyInfosupports_streaming(aiogram.types.input paid media video.InputPaidi
                                                                                                   attribute), 193
              attribute), 135
story (aiogram.types.message.Message attribute), 206
                                                                                     switch_inline_query
Story (class in aiogram.types.story), 273
                                                                                                   (aiogram.types.inline keyboard button.InlineKeyboardButton
stream_content() (aiogram.client.session.base.BaseSession
                                                                                                   attribute), 183
                                                                                     switch_inline_query_chosen_chat
              method), 16
street_line1(aiogram.types.shipping_address.ShippingAddress
                                                                                                   (aiogram.types.inline_keyboard_button.InlineKeyboardButton
              attribute), 343
                                                                                                   attribute), 184
street_line2(aiogram.types.shipping_address.ShippingAskdritxsh_inline_query_current_chat
                                                                                                   (aiogram.types.inline_keyboard_button.InlineKeyboardButton
              attribute), 343
STRIKETHROUGH (aiogram.enums.message_entity_type.MessageEntityatypibute), 183
             attribute), 558
                                                                                     switch_pm_parameter
Strikethrough (class in aiogram.utils.formatting), 667
                                                                                                   (aiogram.methods.answer_inline_query.AnswerInlineQuery
subscription_period
                                                                                                   attribute), 523
              (aiogram.methods.create_chat_subscription_inviteswirtkc6reputeCentSudiogriptionethoitesLinkswer_inline_query.AnswerInlineQu
              attribute), 398
                                                                                                   attribute), 523
subscription_period
                                                                                    SwitchInlineQueryChosenChat
                                                                                                                                                 (class
                                                                                                                                                                   in
              (aiogram.types.chat_invite_link.ChatInviteLink
                                                                                                   aiogram.types.switch inline query chosen chat),
              attribute), 62
{\tt subscription\_price} \ (a iogram. methods. create\_chat\_subscription\_invite\_link. CreateChatSubscriptionInviteLink) \\
              attribute), 398
subscription\_price (a iogram.types.chat\_invite\_link.Chat \verb|FilterLink|\_ADS| (a iogram.enums.transaction\_partner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.TransactionPartner\_type.Transac
              attribute), 62
                                                                                                   attribute), 561
SUCCEEDED (aiogram.enums.revenue_withdrawal_state_type_BleegrealViplosmerclSchtaFyee_id
              attribute), 560
                                                                                                   (aiogram.methods.refund_star_payment.RefundStarPayment
SUCCESSFUL_PAYMENT (aiogram.enums.content_type.ContentType
                                                                                                   attribute), 536
             attribute), 550
                                                                                     telegram_payment_charge_id
successful_payment (aiogram.types.message.Message
                                                                                                   (aiogram.types.refunded_payment.RefundedPayment
              attribute), 208
                                                                                                   attribute), 340
SuccessfulPayment
                                                   (class
                                                                                  telegram_payment_charge_id
              aiogram.types.successful_payment), 346
                                                                                                   (aiogram.types.successful_payment.SuccessfulPayment
suggested_tip_amounts
                                                                                                   attribute), 346
              (aiogram.methods.create_invoice_link.CreateInvotaeLeighramAPIError, 626
             attribute), 533
                                                                                    TelegramAPIServer (class in aiogram.client.telegram),
suggested_tip_amounts
              (aiogram.methods.send_invoice.SendInvoice
                                                                                     TelegramBadRequest, 626
              attribute), 538
                                                                                    TelegramConflictError, 626
suggested_tip_amounts
                                                                                    TelegramEntityTooLarge, 626
              (aiogram.types.input_invoice_message_content.Inpullegriant/bashagedeontenor, 626
              attribute), 328
                                                                                     TelegramMigrateToChat, 626
SUPERGROUP
                             (aiogram.enums.chat type.ChatType
                                                                                    TelegramNetworkError, 626
              attribute), 549
                                                                                    TelegramNotFound, 626
SUPERGROUP_CHAT_CREATED
                                                                                     TelegramRetryAfter, 626
              (aiogram.enums.content_type.ContentType
                                                                                    TelegramServerError, 626
             attribute), 550
                                                                                    TelegramUnauthorizedError, 626
supergroup_chat_created
                                                                                     TEMPORARY_REGISTRATION
              (aiogram.types.message.Message
                                                                   attribute),
                                                                                                   (aiogram.enums.encrypted_passport_element.EncryptedPassport
                                                                                                   attribute), 555
supports_inline_queries (aiogram.types.user.User
                                                                                   TEXT
                                                                                                 (aiogram.enums.content_type.ContentType
             attribute), 276
                                                                                                   tribute), 549
attribute), 477
                                                                                                   erty), 580
```

attribute), 479

```
attribute), 384
                                                                                                     attribute), 381
text(aiogram.methods.edit_message_text.EditMessageTextthumbnail
                                                                                                           (aiogram.types.animation.Animation
              attribute), 517
                                                                                                     tribute), 20
text (aiogram.methods.send_message.SendMessage at-
                                                                                      thumbnail (aiogram.types.audio.Audio attribute), 21
              tribute), 462
                                                                                      thumbnail
                                                                                                            (aiogram.types.document.Document at-
text (aiogram.types.game.Game attribute), 364
                                                                                                     tribute), 134
text (aiogram.types.inline_keyboard_button.InlineKeyboardButtonail (aiogram.types.input_media_animation.InputMediaAnimation
              attribute), 183
                                                                                                     attribute), 186
text (aiogram.types.inline_query_results_button.InlineQuertyManbalesButtaiogram.types.input_media_audio.InputMediaAudio
              attribute), 325
                                                                                                     attribute), 187
text (aiogram.types.input_poll_option.InputPollOption
                                                                                      thumbnail (aiogram.types.input_media_document.InputMediaDocument
              attribute), 193
                                                                                                     attribute), 188
                                                                                      thumbnail (aiogram.types.input_media_video.InputMediaVideo
          (aiogram.types.keyboard_button.KeyboardButton
              attribute), 194
                                                                                                     attribute), 190
text
             (aiogram.types.menu_button.MenuButton
                                                                                      thumbnail (aiogram.types.input_paid_media_video.InputPaidMediaVideo
                                                                                                     attribute), 192
              tribute), 201
text (aiogram.types.menu_button_web_app.MenuButtonWebAppbnail (aiogram.types.sticker.Sticker attribute), 335
                                                                                      thumbnail
                                                                                                           (aiogram.types.sticker_set.StickerSet
              attribute), 203
text (aiogram.types.message.Message attribute), 206
                                                                                                     tribute), 336
text (aiogram.types.poll_option.PollOption attribute),
                                                                                      thumbnail (aiogram.types.video.Video attribute), 280
                                                                                      thumbnail (aiogram.types.video_note.VideoNote at-
text (aiogram.types.text_quote.TextQuote attribute), 275
                                                                                                     tribute), 282
             (aiogram.utils.callback_answer.CallbackAnswer
                                                                                      thumbnail_height(aiogram.types.inline_query_result_article.InlineQuery_
              property), 661
                                                                                                     attribute), 288
Text (class in aiogram.utils.formatting), 665
                                                                                      thumbnail_height(aiogram.types.inline_query_result_contact.InlineQue
text_entities (aiogram.types.game.Game attribute),
                                                                                                     attribute), 308
                                                                                      thumbnail_height(aiogram.types.inline_query_result_document.InlineQ
text_entities(aiogram.types.input_poll_option.InputPollOption attribute), 310
                                                                                      thumbnail\_height ({\it aiogram.types.inline\_query\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_location.InlineQuery\_result\_l
              attribute), 193
text_entities (aiogram.types.poll_option.PollOption
                                                                                                     attribute), 315
              attribute), 267
                                                                                      thumbnail_height(aiogram.types.inline_query_result_venue.InlineQuer
TEXT_LINK (aiogram.enums.message_entity_type.MessageEntityType attribute), 321
                                                                                      thumbnail_mime_type
              attribute), 558
TEXT_MENTION (aiogram.enums.message_entity_type.MessageEntityTypeiogram.types.inline_query_result_gif.InlineQueryResultGif
              attribute), 558
                                                                                                     attribute), 313
text_parse_mode(aiogram.types.input_poll_option.InputRahbahbriail_mime_type
              attribute), 193
                                                                                                     (aiogram.types.inline_query_result_mpeg4_gif.InlineQueryResul
TextLink (class in aiogram.utils.formatting), 667
                                                                                                     attribute), 317
TextMention (class in aiogram.utils.formatting), 667
                                                                                      thumbnail_url(aiogram.types.inline_query_result_article.InlineQueryRe
TextQuote (class in aiogram.types.text_quote), 275
                                                                                                     attribute), 288
THB (aiogram.enums.currency.Currency attribute), 553
                                                                                      thumbnail_url(aiogram.types.inline_query_result_contact.InlineQueryR
theme_name (aiogram.types.background_type_chat_theme.BackgroundffyibaCdigtTNeme
              attribute), 23
                                                                                      thumbnail_url(aiogram.types.inline_query_result_document.InlineQuery
\verb|thumbnail| (a iogram.methods.send\_animation.SendAnimation|
                                                                                                     attribute), 310
                                                                                      thumbnail_url(aiogram.types.inline_query_result_gif.InlineQueryResult
              attribute), 443
thumbnail (aiogram.methods.send_audio.SendAudio at-
                                                                                                     attribute), 312
              tribute), 446
                                                                                      thumbnail_url(aiogram.types.inline_query_result_location.InlineQuery|
thumbnail ({\it aiogram.methods.send\_document.SendDocument}
                                                                                                     attribute), 315
              attribute), 454
                                                                                       thumbnail_url(aiogram.types.inline_query_result_mpeg4_gif.InlineQuery_
thumbnail (aiogram.methods.send_video.SendVideo at-
                                                                                                     attribute), 317
              tribute), 477
                                                                                      thumbnail_url(aiogram.types.inline_query_result_photo.InlineQueryRes
thumbnail ({\it aiogram.methods.send\_video\_note.SendVideoNote}
                                                                                                     attribute), 319
```

text (aiogram.methods.answer\_callback\_query.AnswerCallback\_query.AnswerCallback\_duery.AnswerCallback\_query.AnswerC

798 Index

 $thumbnail\_url (\it aiogram.types.inline\_query\_result\_venue.InlineQueryResult\_v$ 

```
attribute), 321
                                                                                                                                         title(aiogram.types.inline_query_result_cached_voice.InlineQueryResult
thumbnail_url(aiogram.types.inline_query_result_video.InlineQuerytReisuleVideo
                      attribute), 323
                                                                                                                                         title(aiogram.types.inline_query_result_document.InlineQueryResultDoc
thumbnail_width(aiogram.types.inline_query_result_article.Inline@tteibuResultADticle
                      attribute), 288
                                                                                                                                          title(aiogram.types.inline_query_result_gif.InlineQueryResultGif
thumbnail_width(aiogram.types.inline query result contact.Inline@nrebynRe\sidltGontact
                      attribute), 308
                                                                                                                                          {\tt title}\ (aiogram.types.inline\_query\_result\_location.InlineQueryResultLocation)
thumbnail_width(aiogram.types.inline_query_result_document.InlimatQhuery)ResultDocument
                      attribute), 310
                                                                                                                                          title(aiogram.types.inline_query_result_mpeg4_gif.InlineQueryResultMy
thumbnail_width(aiogram.types.inline_query_result_location.Inline@nibryR);sidliLocation
                      attribute), 315
                                                                                                                                         \verb|title| (a iogram.types.in line\_query\_result\_photo.In lineQueryResultPhoto|)|
thumbnail_width(aiogram.types.inline_query_result_venue.InlineQuery)liMenue
                      attribute), 321
                                                                                                                                          title(aiogram.types.inline_query_result_venue.InlineQueryResultVenue
time_zone_name (aiogram.types.business_opening_hours.BusinessOpptaniibustle) nurs.BusinessOpptaniibustle) nurs.BusinessOpptaniibustl
                      attribute), 33
                                                                                                                                          \verb|title| (a iogram.types.in line\_query\_result\_video.In lineQueryResultVideo|)|
timeout (aiogram.methods.get_updates.GetUpdates at-
                                                                                                                                                                 attribute), 323
                      tribute), 542
                                                                                                                                         title(aiogram.types.inline_query_result_voice.InlineQueryResultVoice
title(aiogram.methods.create_invoice_link.CreateInvoiceLink
                                                                                                                                                                 attribute), 324
                                                                                                                                          title(aiogram.types.input_invoice_message_content.InputInvoiceMessage
                       attribute), 533
{\tt title}\ (aiogram.methods.create\_new\_sticker\_set.CreateNewStickerSet uttribute), 327
                      attribute), 367
                                                                                                                                          title(aiogram.types.input_media_audio.InputMediaAudio
                      (aiogram.methods.send_audio.SendAudio
                                                                                                                                                                 attribute), 188
                      tribute), 446
                                                                                                                                          title(aiogram.types.input_venue_message_content.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputVenueMessageContent.InputContent.InputContent.InputContent.InputContent.InputContent.InputContent.InputContent.InputContent.InputContent.InputContent.InputContent.InputContent.InputContent.InputContent.InputContent.InputContent.InputContent.InputContent.InputContent.InputContent.InputContent.InputContent.InputContent.InputContent.InputContent.InputContent.InputContent.InputContent.InputContent.InputContent.InputCon
title (aiogram.methods.send_invoice.SendInvoice at-
                                                                                                                                                                 attribute), 332
                      tribute), 538
                                                                                                                                         title (aiogram.types.invoice.Invoice attribute), 337
title
                      (aiogram.methods.send_venue.SendVenue
                                                                                                                                         title
                                                                                                                                                              (aiogram.types.shipping_option.ShippingOption
                      tribute), 473
                                                                                                                                                                 attribute), 343
{\tt title} \ (a iogram.methods.set\_chat\_title.SetChatTitle \ at-
                                                                                                                                         title (aiogram.types.sticker_set.StickerSet attribute),
                      tribute), 491
title (aiogram.methods.set_sticker_set_title.SetStickerSetTttle1e (aiogram.types.venue.Venue attribute), 279
                      attribute), 382
                                                                                                                                          title (aiogram.utils.web_app.WebAppChat attribute),
title (aiogram.types.audio.Audio attribute), 20
title (aiogram.types.business_intro.BusinessIntro at-
                                                                                                                                         TJS (aiogram.enums.currency.Currency attribute), 553
                                                                                                                                          TokenBasedRequestHandler
                      tribute), 32
                                                                                                                                                                                                                                       (class
                                                                                                                                                                                                                                                                         in
title (aiogram.types.chat.Chat attribute), 36
                                                                                                                                                                 aiogram.webhook.aiohttp_server), 631
title (aiogram.types.chat_full_info.ChatFullInfo
                                                                                                                                         top_color (aiogram.types.background_fill_gradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGradient.BackgroundFillGrad
                                                                                                                                                                 attribute), 22
title
                       (aiogram.types.chat_shared.ChatShared
                                                                                                                                         {\tt TopicIconColor}
                                                                                                                                                                                                                                                                         in
                                                                                                                              at-
                                                                                                                                                                                                                        (class
                       tribute), 132
                                                                                                                                                                 aiogram.enums.topic_icon_color), 561
title (aiogram.types.game.Game attribute), 364
                                                                                                                                          total_amount (aiogram.types.invoice.Invoice attribute),
title(aiogram.types.inline_query_result_article.InlineQueryResultAfficle
                       attribute), 287
                                                                                                                                          total_amount(aiogram.types.pre_checkout_query.PreCheckoutQuery
title(aiogram.types.inline_query_result_audio.InlineQueryResultAudixibute), 339
                      attribute), 289
                                                                                                                                          total\_amount(aiogram.types.refunded\_payment.RefundedPayment
title (aiogram.types.inline_query_result_cached_document.InlineQuatryRused)t&WhedDocument
                                                                                                                                          total_amount(aiogram.types.successful_payment.SuccessfulPayment
                      attribute), 294
title (aiogram.types.inline_query_result_cached_gif.InlineQueryResultChade)dGiff
                      attribute), 296
                                                                                                                                          total_count (aiogram.types.reaction_count.ReactionCount
title (aiogram.types.inline_query_result_cached_mpeg4_gif.InlineQuery|Rues)ulfCachedMpeg4Gif
                                                                                                                                          {\tt total\_count} \ (a iogram. types. user\_profile\_photos. User Profile Photos
                      attribute), 298
title(aiogram.types.inline_query_result_cached_photo.InlineQueryRtesilt(Ca);ch@dPhoto
                      attribute), 300
                                                                                                                                          total_voter_count (aiogram.types.poll.Poll attribute),
title (aiogram.types.inline_query_result_cached_video.InlineQueryR65ultCachedVideo
                                                                                                                                         TransactionPartner
                      attribute), 304
                                                                                                                                                                                                                              (class
                                                                                                                                                                                                                                                                         in
```

```
aiogram.types.transaction_partner), 347
                                                                                                 attribute), 29
TransactionPartnerFragment
                                                                             in type (aiogram.types.bot_command_scope_chat_member.BotCommandScop
                                                          (class
                                                                                                 attribute), 29
             aiogram.types.transaction_partner_fragment),
                                                                                   type (aiogram.types.bot_command_scope_default.BotCommandScopeDefa
TransactionPartnerOther
                                                       (class
                                                                             in
                                                                                                 attribute), 30
                                                                                   type (aiogram.types.chat.Chat attribute), 36
             aiogram.types.transaction_partner_other),
                                                                                   type (aiogram.types.chat_background.ChatBackground
TransactionPartnerTelegramAds
                                                             (class
                                                                             in
                                                                                                 attribute), 53
                                                                                               (aiogram.types.chat\_full\_info.ChatFullInfo
             aiogram.types.transaction_partner_telegram_ads)type
                                                                                                 tribute), 58
TransactionPartnerType
                                                      (class
                                                                                   type (aiogram.types.encrypted_passport_element.EncryptedPassportEleme
             aiogram.enums.transaction_partner_type),
                                                                                                 attribute), 350
                                                                                   type (aiogram.types.inline_query_result_article.InlineQueryResultArticle
TransactionPartnerUser
                                                      (class
                                                                             in
                                                                                                 attribute), 287
             aiogram.types.transaction_partner_user),
                                                                                   {\tt type} \ (aiogram.types.inline\_query\_result\_audio.InlineQueryResultAudio
              349
                                                                                                 attribute), 289
transactions (aiogram.types.star_transactions.StarTransactions.StarTransactions.inline_query_result_cached_audio.InlineQueryResult
             attribute), 345
                                                                                                 attribute), 291
translation (aiogram.types.encrypted_passport_element.Eypps/triadflussptypteSlemlint_query_result_cached_document.InlineQueryRe
             attribute), 351
                                                                                                 attribute), 294
TRANSLATION_FILE (aiogram.enums.passport_element_erroypts/paibgssport/besinehtler.queliyperesult_cached_gif.InlineQueryResultCac
             attribute), 559
                                                                                                 attribute), 295
TRANSLATION_FILES (aiogram.enums.passport_element_erropet(wpio@rassportpEsianleneEquerflypresult_cached_mpeg4_gif.InlineQueryR
                                                                                                 attribute), 298
             attribute), 559
attribute), 267
                                                                                                 attribute), 300
TRY (aiogram.enums.currency.Currency attribute), 553
                                                                                   type (aiogram.types.inline_query_result_cached_sticker.InlineQueryResult
TTD (aiogram.enums.currency.Currency attribute), 553
                                                                                                 attribute), 302
TWD (aiogram.enums.currency.Currency attribute), 553
                                                                                   type (aiogram.types.inline_query_result_cached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryResultCached_video.InlineQueryRe
type (aiogram.methods.send_poll.SendPoll attribute),
                                                                                                 attribute), 304
                                                                                    type (aiogram.types.inline_query_result_cached_voice.InlineQueryResultC
type (aiogram.types.background_fill_freeform_gradient.BackgroundFilltFileufo);nGradient
             attribute), 21
                                                                                   type (aiogram.types.inline_query_result_contact.InlineQueryResultContact
type (aiogram.types.background_fill_gradient.BackgroundFillGradientttribute), 307
              attribute), 22
                                                                                   type (aiogram.types.inline_query_result_document.InlineQueryResultDocument.
{\tt type} \ (aiogram. types. background\_fill\_solid. Background FillSolid
                                                                                                 attribute), 309
             attribute), 22
                                                                                   type (aiogram.types.inline_query_result_game.InlineQueryResultGame
type (aiogram.types.background_type_chat_theme.BackgroundTypeChttrTbueta); 311
             attribute), 23
                                                                                    type (aiogram.types.inline_query_result_gif.InlineQueryResultGif
type (aiogram.types.background_type_fill.BackgroundTypeFill
                                                                                                 attribute), 312
                                                                                   type (aiogram.types.inline_query_result_location.InlineQueryResultLocati
             attribute), 23
{\tt type} \ (aiogram. types. background\_type\_pattern. Background Type Patter \textit{\textbf{n}} ttribute), 314
                                                                                   type (aiogram.types.inline_query_result_mpeg4_gif.InlineQueryResultMpe
              attribute), 24
type (aiogram.types.background_type_wallpaper.BackgroundTypeWallpaiberte), 317
                                                                                   {\tt type} \ (aiogram.types.inline\_query\_result\_photo.InlineQueryResultPhoto
             attribute), 25
type (aiogram.types.bot_command_scope_all_chat_administrators.Bott@bbuta)nd$8opeAllChatAdministrators
             attribute), 27
                                                                                   type (aiogram.types.inline_query_result_venue.InlineQueryResultVenue
type (aiogram.types.bot_command_scope_all_group_chats.BotCommatixiBoxop);AllGroupChats
             attribute), 27
                                                                                   type (aiogram.types.inline_query_result_video.InlineQueryResultVideo
type (aiogram.types.bot_command_scope_all_private_chats.BotCommutaridbSutoneAllPrivateChats
                                                                                   type (aiogram.types.inline_query_result_voice.InlineQueryResultVoice
             attribute), 28
type (aiogram.types.bot_command_scope_chat.BotCommandScopeChattribute), 324
             attribute), 28
                                                                                   {\tt type} \ (aiogram. types. input\_media\_animation. Input Media Animation
```

type (aiogram.types.bot command scope chat administrators.BotCattmibutkScopeChatAdministrators

```
type (aiogram.types.input_media_audio.InputMediaAudio_type (aiogram.types.passport_element_error_unspecified.PassportElement
                       attribute), 187
                                                                                                                                                                   attribute), 359
type (aiogram.types.input media document.InputMediaDotypentiogram.types.poll.Poll attribute), 266
                                                                                                                                            type (aiogram.types.reaction_count.ReactionCount at-
                       attribute), 188
type (aiogram.types.input_media_photo.InputMediaPhoto
                                                                                                                                                                   tribute), 268
                      attribute), 189
                                                                                                                                            type (aiogram.types.reaction type custom emoji.ReactionTypeCustomEmo
type (aiogram.types.input media video.InputMediaVideo
                                                                                                                                                                   attribute), 268
                                                                                                                                            \verb"type" (a iogram.types.reaction\_type\_emoji.ReactionTypeEmoji"
                       attribute), 190
type (aiogram.types.input paid media photo.InputPaidMediaPhoto attribute), 269
                       attribute), 192
                                                                                                                                            type (aiogram.types.reaction_type_paid.ReactionTypePaid
type (aiogram.types.input_paid_media_video.InputPaidMediaVideo attribute), 269
                       attribute), 192
                                                                                                                                            type (aiogram.types.revenue_withdrawal_state_failed.RevenueWithdrawals
type (aiogram.types.keyboard_button_poll_type.KeyboardButtonPollTypeibute), 341
                      attribute), 195
                                                                                                                                            type (aiogram.types.revenue_withdrawal_state_pending.RevenueWithdraw
                     (aiogram.types.menu_button.MenuButton
                                                                                                                                                                   attribute), 341
type
                       tribute), 201
                                                                                                                                            type (aiogram.types.revenue_withdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_state_succeeded.RevenueWithdrawal_s
type (aiogram.types.menu_button_commands.MenuButtonCommandsattribute), 342
                      attribute), 202
                                                                                                                                            type (aiogram.types.sticker.Sticker attribute), 334
type (aiogram.types.menu_button_default.MenuButtonDefaultpe (aiogram.types.transaction_partner_fragment.TransactionPartnerFra
                       attribute), 202
                                                                                                                                                                    attribute), 347
type (aiogram.types.menu_button_web_app.MenuButtonWebypp(aiogram.types.transaction_partner_other.TransactionPartnerOther
                       attribute), 203
                                                                                                                                                                   attribute), 348
type (aiogram.types.message_entity.MessageEntity at- type(aiogram.types.transaction_partner_telegram_ads.TransactionPartne
                       tribute), 257
                                                                                                                                                                   attribute), 348
type (aiogram.types.message_origin_channel.MessageOrigtry@ka(uniokram.types.transaction_partner_user.TransactionPartnerUser
                       attribute), 259
                                                                                                                                                                   attribute), 349
type (aiogram.types.message_origin_chat.MessageOriginChype (aiogram.utils.web_app.WebAppChat attribute),
                       attribute), 259
type (aiogram.types.message_origin_hidden_user.Message OYPRINHiddaib/seam.enums.chat_action.ChatAction
                       attribute), 260
                                                                                                                                                                    tribute), 547
type(aiogram.types.message_origin_user.MessageOriginUtypping() (aiogram.utils.chat_action.ChatActionSender
                       attribute), 261
                                                                                                                                                                    class method), 652
type (aiogram.types.paid_media_photo.PaidMediaPhoto TZS (aiogram.enums.currency.Currency attribute), 553
                       attribute), 263
type (aiogram.types.paid_media_preview.PaidMediaPrevie
                       attribute), 264
                                                                                                                                            UAH (aiogram.enums.currency.Currency attribute), 553
type (aiogram.types.paid media video.PaidMediaVideo
                                                                                                                                           UGX (aiogram.enums.currency.Currency attribute), 554
                       attribute), 264
                                                                                                                                           unban() (aiogram.types.chat.Chat method), 49
type (aiogram.types.passport_element_error_data_field.PassportElement_error_DataField (aiogram.types.chat.Chat
                       attribute), 353
                                                                                                                                                                   method), 41
type (aiogram.types.passport_element_error_file.PassportElementError_file.PassportElementError_file.PassportElementError_file.PassportElementError_file.PassportElementError_file.PassportElementError_file.PassportElementError_file.PassportElementError_file.PassportElementError_file.PassportElementError_file.PassportElementError_file.PassportElementError_file.PassportElementError_file.PassportElementError_file.PassportElementError_file.PassportElementError_file.PassportElementError_file.PassportElementError_file.PassportElementError_file.PassportElementError_file.PassportElementError_file.PassportElementError_file.PassportElementError_file.PassportElementError_file.PassportElementError_file.PassportElementError_file.PassportElementError_file.PassportElementError_file.PassportElementError_file.PassportElementError_file.PassportElementError_file.PassportElementError_file.PassportElementError_file.PassportElementError_file.PassportElementError_file.PassportElementError_file.PassportElementError_file.PassportElementError_file.PassportElementError_file.PassportElementError_file.PassportElementError_file.PassportElementError_file.PassportElementError_file.PassportElementError_file.PassportElementError_file.PassportElementError_file.PassportElementError_file.PassportError_file.PassportError_file.PassportError_file.PassportError_file.PassportError_file.PassportError_file.PassportError_file.PassportError_file.PassportError_file.PassportError_file.PassportError_file.PassportError_file.PassportError_file.PassportError_file.PassportError_file.PassportError_file.PassportError_file.PassportError_file.PassportError_file.PassportError_file.PassportError_file.PassportError_file.PassportError_file.PassportError_file.PassportError_file.PassportError_file.PassportError_file.PassportError_file.PassportError_file.PassportError_file.PassportError_file.PassportError_file.PassportError_file.PassportError_file.PassportError_file.PassportError_file.PassportError_file.PassportError_file.PassportError_file.PassportError_file.PassportError_fi
                                                                                                                                                                                                                             (class
                                                                                                                                                                                                                                                                             in
                       attribute), 353
                                                                                                                                                                   aiogram.methods.unban_chat_member),
type (aiogram.types.passport_element_error_files.PassportElementError_Files
                       attribute), 354
                                                                                                                                            UnbanChatSenderChat
                                                                                                                                                                                                                                   (class
                                                                                                                                                                                                                                                                             in
type (aiogram.types.passport_element_error_front_side.PassportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElement_error_front_side.passportElem
                       attribute), 355
type (aiogram.types.passport_element_error_reverse_side.PassportfelamentErrorBayerseSide
                      attribute), 356
                                                                                                                                                                    (aiogram.types.giveaway_completed.GiveawayCompleted
type (aiogram.types.passport_element_error_selfie.PassportElementEnropSelfie 141
                       attribute), 357
                                                                                                                                           unclaimed_prize_count
type (aiogram.types.passport_element_error_translation_file.PassportElementEypes.Types.Types.Wileners.GiveawayWinners
                       attribute), 357
                                                                                                                                                                   attribute), 142
attribute), 358
                                                                                                                                                                   attribute), 558
```

```
Underline (class in aiogram.utils.formatting), 667
                                                       UpdateType (class in aiogram.enums.update type), 562
UnhideGeneralForumTopic
                                     (class
                                                   in UpdateTypeLookupError, 362
         aiogram.methods.unhide general forum topic), UPLOAD_DOCUMENT (aiogram.enums.chat action.ChatAction
                                                                 attribute), 548
UNKNOWN (aiogram.enums.content type.ContentType at-
                                                       upload_document() (aiogram.utils.chat_action.ChatActionSender
         tribute), 549
                                                                class method), 652
unpack() (aiogram.filters.callback data.CallbackData
                                                       UPLOAD_PHOTO (aiogram.enums.chat action.ChatAction
         class method), 586
                                                                 attribute), 547
unpin() (aiogram.types.message.Message method), 254
                                                       upload_photo() (aiogram.utils.chat action.ChatActionSender
unpin_all_general_forum_topic_messages()
                                                                class method), 652
         (aiogram.types.chat.Chat method), 51
                                                       UPLOAD_VIDEO (aiogram.enums.chat_action.ChatAction
unpin_all_messages()
                                                                attribute), 547
                             (aiogram.types.chat.Chat
         method), 45
                                                       upload_video() (aiogram.utils.chat_action.ChatActionSender
unpin_message() (aiogram.types.chat.Chat method),
                                                                class method), 653
                                                       {\tt UPLOAD\_VIDEO\_NOTE} \ (aiogram.enums.chat\_action.ChatAction
UnpinAllChatMessages
                                   (class
                                                   in
                                                                 attribute), 548
         aiogram.methods.unpin_all_chat_messages),
                                                       upload_video_note()
                                                                (aiogram.utils.chat\_action.ChatActionSender
UnpinAllForumTopicMessages
                                                                class method), 653
                                                   in
                                      (class
         aiogram.methods.unpin all forum topic message SPLOAD_VOICE (aiogram.enums.chat action.ChatAction
                                                                attribute), 548
UnpinAllGeneralForumTopicMessages (class
                                                   in upload_voice() (aiogram.utils.chat_action.ChatActionSender
         aiogram.methods.unpin_all_general_forum_topic_messagesolass method), 653
         504
                                                       UploadStickerFile
UnpinChatMessage
                                                                 aiogram.methods.upload sticker file), 383
                                (class
                                                   in
                                                       {\tt URL}\,(aiogram.enums.message\_entity\_type.MessageEntityType
         aiogram.methods.unpin chat message), 505
unrestrict_boost_count
                             (aiogram.types.chat.Chat
                                                                attribute), 557
         attribute), 40
                                                       \verb"url" (a iogram.methods.answer\_callback\_query. AnswerCallbackQuery
unrestrict_boost_count
                                                                attribute), 384
                                                              (aiogram.methods.set\_webhook.SetWebhook
         (aiogram.types.chat_full_info.ChatFullInfo
                                                       url
         attribute), 60
                                                                tribute), 544
UNSPECIFIED (aiogram.enums.passport_element_error_typastPaksipogttbilettyepetFirtlomTykeyboard_button.InlineKeyboardButton
         attribute), 559
                                                                attribute), 183
UnsupportedKeywordArgument, 626
                                                       \verb"url" (a iogram.types.in line\_query\_result\_article. In lineQueryResultArticle
until_date(aiogram.methods.ban chat member.BanChatMember attribute), 288
         attribute), 387
                                                       url (aiogram.types.link_preview_options.LinkPreviewOptions
until_date(aiogram.methods.restrict chat member.RestrictChatMeanthrebrute), 199
         attribute), 440
                                                       url (aiogram.types.login_url.LoginUrl attribute), 200
until_date(aiogram.types.chat_member_banned.ChatMewablerBannedaiogram.types.message_entity.MessageEntity
         attribute), 106
                                                                attribute), 257
until_date(aiogram.types.chat member member.ChatMeurtheynditegnber.types.revenue withdrawal state succeeded.RevenueWithdrav
         attribute), 107
                                                                 attribute), 342
until_date (aiogram.types.chat member restricted.ChatMenhteentexpes.user.User property), 276
         attribute), 109
                                                       url (aiogram.types.web_app_info.WebAppInfo attribute),
          (aiogram.types.error_event.ErrorEvent
update
                                                  at-
                                                              (aiogram.types.webhook_info.WebhookInfo
         tribute), 625
                                                       url
Update (class in aiogram.types.update), 360
                                                                tribute), 363
update_data()
                      (aiogram.fsm.scene.SceneWizard
                                                       url
                                                                (aiogram.utils.callback_answer.CallbackAnswer
         method), 621
                                                                property), 661
update_data() (aiogram.fsm.storage.base.BaseStorage
                                                       Url (class in aiogram.utils.formatting), 666
         method), 603
                                                       URLInputFile (class in aiogram.types.input_file), 185,
update_handler_flags() (aiogram.filters.base.Filter
         method), 590
                                                       USD (aiogram.enums.currency.Currency attribute), 554
update_id (aiogram.types.update.Update attribute), 361
                                                      use_independent_chat_permissions
```

```
(aiogram.methods.restrict_chat_member.RestrictChatMembattribute), 386
              attribute), 439
                                                                                     user_id(aiogram.methods.create_new_sticker_set.CreateNewStickerSet
use_independent_chat_permissions
                                                                                                    attribute), 366
              (aiogram.methods.set_chat_permissions.SetChatPasnix_sidn(aiogram.methods.decline_chat_join_request.DeclineChatJoinRed
              attribute), 488
                                                                                                    attribute), 401
USER (aiogram.enums.message origin type.MessageOriginTisper_id (aiogram.methods.get chat member.GetChatMember
                                                                                                    attribute), 418
              attribute), 558
USER (aiogram.enums.transaction_partner_type.TransactionPsetneirTxpiogram.methods.get_game_high_scores.GetGameHighScores
              attribute), 561
                                                                                                    attribute), 526
user (aiogram.types.business_connection.BusinessConnectionser_id (aiogram.methods.get_user_chat_boosts.GetUserChatBoosts
              attribute), 31
                                                                                                    attribute), 429
user (aiogram.types.chat_boost_source_gift_code.ChatBoostSonracCiftCynden.methods.get_user_profile_photos.GetUserProfilePhotos
              attribute), 56
                                                                                                    attribute), 429
user (aiogram.types.chat_boost_source_giveaway.ChatBoostsenradGiviergwarn.methods.promote_chat_member.PromoteChatMember
              attribute), 56
                                                                                                    attribute), 435
user (aiogram.types.chat_boost_source_premium.ChatBooxtSeurcielP(aioignum.methods.refund_star_payment.RefundStarPayment
              attribute), 57
                                                                                                    attribute), 536
user (aiogram.types.chat_member_administrator.ChatMember\did\administrator.methods.replace_sticker_in_set.ReplaceStickerInSet
              attribute), 104
                                                                                                    attribute), 372
user (aiogram.types.chat_member_banned.ChatMemberBauser_id (aiogram.methods.restrict_chat_member.RestrictChatMember
              attribute), 106
                                                                                                    attribute), 439
user (aiogram.types.chat_member_left.ChatMemberLeft user_id (aiogram.methods.set_chat_administrator_custom_title.SetChatAc
                                                                                                    attribute), 484
              attribute), 106
user (aiogram.types.chat member member.ChatMemberMersber_id (aiogram.methods.set game score.SetGameScore
              attribute), 107
                                                                                                    attribute), 529
user (aiogram.types.chat_member_owner.ChatMemberOwner_id (aiogram.methods.set_passport_data_errors.SetPassportDataErro
              attribute), 107
                                                                                                    attribute), 546
user (aiogram.types.chat_member_restricted.ChatMemberRessenicted (aiogram.methods.set_sticker_set_thumbnail.SetStickerSetThumbr
              attribute), 108
                                                                                                    attribute), 380
user (aiogram.types.game_high_score.GameHighScore user_id (aiogram.methods.unban_chat_member.UnbanChatMember
              attribute), 365
                                                                                                    attribute), 499
user (aiogram.types.message_entity.MessageEntity at- user_id(aiogram.methods.upload_sticker_file.UploadStickerFile
              tribute), 257
                                                                                                    attribute), 383
user (aiogram.types.message_reaction_updated.MessageReastion_liphlaniaegram.types.bot_command_scope_chat_member.BotCommand
              attribute), 262
                                                                                                    attribute), 30
user (aiogram.types.poll_answer.PollAnswer attribute), user_id (aiogram.types.contact.Contact attribute), 133
                                                                                     user_id (aiogram.types.shared_user.SharedUser
user (aiogram.types.transaction_partner_user.TransactionPartnerUseribute), 273
              attribute), 349
                                                                                      user_id (aiogram.types.user_shared.UserShared at-
user (aiogram.utils.web_app.WebAppInitData attribute),
                                                                                                    tribute), 278
                                                                                     user_ids (aiogram.types.users_shared.UsersShared at-
User (class in aiogram.types.user), 275
                                                                                                    tribute), 278
                                                                                     USER_IN_CHAT (aiogram.fsm.strategy.FSMStrategy at-
user_administrator_rights
              (aiogram.types.keyboard_button_request_chat.KeyboardButtcibRte)uestChat
                                                                                     USER_IN_TOPIC (aiogram.fsm.strategy.FSMStrategy at-
              attribute), 196
user_chat_id(aiogram.types.business_connection.BusinessConnecticibute), 603
              attribute), 31
                                                                                     user_is_bot (aiogram.types.keyboard_button_request_user.KeyboardButt
user_chat_id(aiogram.types.chat_join_request.ChatJoinRequest attribute), 197
                                                                                     \verb"user_is_bot" (a iogram. types. keyboard\_button\_request\_users. KeyboardButton\_request\_users. 
              attribute), 62
user_id(aiogram.methods.add_sticker_to_set.AddStickerToSet
                                                                                                    attribute), 198
                                                                                     user_is_premium(aiogram.types.keyboard_button_request_user.Keyboar
              attribute), 365
user_id(aiogram.methods.approve_chat_join_request.ApproveChatJadurRaque,s197
              attribute), 385
                                                                                     user_is_premium(aiogram.types.keyboard_button_request_users.Keyboa
```

attribute), 198

user\_id(aiogram.methods.ban chat member.BanChatMember

USER_SHARED (aiogram.enums.content_type.ContentType attribute), 551	<pre>via_chat_folder_invite_link</pre>
user_shared (aiogram.types.message.Message at- tribute), 210	attribute), 110 via_join_request (aiogram.types.chat_member_updated.ChatMember_
UserChatBoosts (class in	attribute), 110
aiogram.types.user_chat_boosts), 277 username (aiogram.types.chat.Chat attribute), 36	VIDEO (aiogram.enums.content_type.ContentType at- tribute), 549
username (aiogram.types.chat_full_info.ChatFullInfo at- tribute), 59	VIDEO (aiogram.enums.inline_query_result_type.InlineQueryResultType attribute), 555
username (aiogram.types.chat_shared.ChatShared at- tribute), 132	VIDEO (aiogram.enums.input_media_type.InputMediaType attribute), 556
username (aiogram.types.shared_user.SharedUser attribute), 273	VIDEO (aiogram.enums.input_paid_media_type.InputPaidMediaType attribute), 556
username (aiogram.types.user.User attribute), 275 username (aiogram.utils.web_app.WebAppChat at-	VIDEO (aiogram.enums.paid_media_type.PaidMediaType attribute), 558
tribute), 658	VIDEO (aiogram.enums.sticker_format.StickerFormat at-
username (aiogram.utils.web_app.WebAppUser at-	tribute), 560
tribute), 656	video (aiogram.methods.send_video.SendVideo at-
UserProfilePhotos (class in	tribute), 476
aiogram.types.user_profile_photos), 277 users (aiogram.types.users_shared.UsersShared at-	video (aiogram.types.external_reply_info.ExternalReplyInfo attribute), 135
	video (aiogram.types.message.Message attribute), 207 evChenRanitigipum.typaxipedid_media_video.PaidMediaVideo attribute), 264
attribute), 280 USERS_SHARED (aiogram.enums.content_type.ContentType	<i>"</i>
attribute), 550	VIDEO_CHAT_ENDED (aiogram.enums.content_type.ContentType
users_shared (aiogram.types.message.Message at-	attribute), 551
tribute), 208 UserShared (class in aiogram.types.user_shared), 277	video_chat_ended (aiogram.types.message.Message attribute), 210
UsersShared (class in aiogram.types.users_shared), 278	VIDEO_CHAT_PARTICIPANTS_INVITED
UTILITY_BILL (aiogram.enums.encrypted_passport_elem	
attribute), 555	attribute), 551
UYU (aiogram.enums.currency.Currency attribute), 554	video_chat_participants_invited
UZS (aiogram.enums.currency.Currency attribute), 554	(aiogram.types.message.Message attribute), 210
V	VIDEO_CHAT_SCHEDULED
value (aiogram.types.dice.Dice attribute), 133	(aiogram.enums.content_type.ContentType
vcard (aiogram.methods.send_contact.SendContact at-	attribute), 551
tribute), 450	video_chat_scheduled
vcard (aiogram.types.contact.Contact attribute), 133	(aiogram.types.message.Message attribute),
vcard(aiogram.types.inline_query_result_contact.InlineQ attribute), 307	VIDEO_CHAT_STARTED (aiogram.enums.content_type.ContentType
vcard(aiogram.types.input_contact_message_content.Inpu attribute), 326	<pre>video_chat_started (aiogram.types.message.Message</pre>
VENUE (aiogram.enums.content_type.ContentType at-	attribute), 209 video_duration(aiogram.types.inline_query_result_video.InlineQuery
tribute), 549	
VENUE (aiogram.enums.inline_query_result_type.InlineQue attribute), 555	video_file_id(aiogram.types.inline_query_result_cached_video.Inlin
venue (aiogram.types.external_reply_info.ExternalReplyIn	fo attribute), 304 video_height (aiogram.types.inline_query_result_video.InlineQueryRe
attribute), 136	attribute), 323
venue (aiogram.types.message.Message attribute), 207 Venue (class in aiogram.types.venue), 279	VIDEO_NOTE (aiogram.enums.content_type.ContentType
venue (ciass in aiogram.types.venue), 219 via_bot (aiogram.types.message.Message attribute), 206	attribute), 549
via_boc (atogram.types.message.message attribute), 200	video_note(aiogram.methods.send_video_note.SendVideoNote

```
attribute), 479
                                                                                                                                           web_app (aiogram.types.keyboard_button.KeyboardButton
video_note (aiogram.types.external reply info.ExternalReplyInfo attribute), 194
                                                                                                                                          web_app (aiogram.types.menu button.MenuButton at-
                      attribute), 135
video_note (aiogram.types.message.Message attribute),
                                                                                                                                                                  tribute), 202
                                                                                                                                           web_app (aiogram.types.menu_button_web_app.MenuButtonWebApp
video_url(aiogram.types.inline query result video.InlineQueryResadtivViolace), 203
                      attribute), 323
                                                                                                                                           WEB_APP_DATA (aiogram.enums.content type.ContentType
video_width(aiogram.types.inline_query_result_video.InlineQuery RuesultiViale, δ51
                       attribute), 323
                                                                                                                                          web_app_data (aiogram.types.message.Message
VideoChatEnded
                                                                              (class
                                                                                                                                in
                                                                                                                                                                  tribute), 210
                       aiogram.types.video_chat_ended), 280
                                                                                                                                          {\tt web\_app\_name} \ (a iogram. types. write\_access\_allowed. WriteAccessAllowed
VideoChatParticipantsInvited
                                                                                                   (class
                                                                                                                                in
                                                                                                                                                                  attribute), 283
                      aiogram.types.video_chat_participants_invited), web_app_query_id(aiogram.methods.answer_web_app_query.AnswerWe
                       280
                                                                                                                                                                  attribute), 524
VideoChatScheduled
                                                                                     (class
                                                                                                                                 in WebAppChat (class in aiogram.utils.web_app), 657
                       aiogram.types.video_chat_scheduled), 281
                                                                                                                                          WebAppData (class in aiogram.types.web_app_data), 283
VideoChatStarted
                                                                                                                                in WebAppInfo (class in aiogram.types.web_app_info), 283
                                                                                 (class
                                                                                                                                           WebAppInitData (class in aiogram.utils.web app), 655
                       aiogram.types.video chat started), 281
VideoNote (class in aiogram.types.video_note), 281
                                                                                                                                           WebAppUser (class in aiogram.utils.web_app), 656
VIOLET (aiogram.enums.topic icon color.TopicIconColor WebhookInfo (class in aiogram.types.webhook info),
                      attribute), 561
VND (aiogram.enums.currency.Currency attribute), 554
                                                                                                                                           width (aiogram.methods.send animation.SendAnimation
VOICE (aiogram.enums.content_type.ContentType at-
                                                                                                                                                                  attribute), 443
                       tribute), 549
                                                                                                                                          width
                                                                                                                                                                  (aiogram.methods.send video.SendVideo
VOICE (aiogram.enums.inline_query_result_type.InlineQueryResultTypebute), 476
                      attribute), 555
                                                                                                                                          width (aiogram.types.animation.Animation attribute), 19
voice
                       (aiogram.methods.send\_voice.SendVoice
                                                                                                                                          \verb|width| (a iogram.types.input\_media\_animation.InputMediaAnimation|)|
                       tribute), 482
                                                                                                                                                                  attribute), 187
voice (aiogram.types.external_reply_info.ExternalReplyInfwidth (aiogram.types.input_media_video.InputMediaVideo
                       attribute), 135
                                                                                                                                                                  attribute), 191
voice (aiogram.types.message.Message attribute), 207
                                                                                                                                          width(aiogram.types.input_paid_media_video.InputPaidMediaVideo
Voice (class in aiogram.types.voice), 282
                                                                                                                                                                  attribute), 193
voice_duration(aiogram.types.inline_query_result_voicewlindintarQuiogyRestylpeoipaid_media_preview.PaidMediaPreview
                       attribute), 325
                                                                                                                                                                  attribute), 264
voice_file_id (aiogram.types.inline query result cachedrivatibe.liding@umertyResulh@tacheid&/FikotoSize attribute),
                      attribute), 306
voice_url (aiogram.types.inline query result voice.Inline@ude\nRaiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nraiog\notate\nra
                       attribute), 324
                                                                                                                                          width (aiogram.types.video.Video attribute), 279
voter_chat (aiogram.types.poll_answer.PollAnswer at-
                                                                                                                                          winner_count (aiogram.types.giveaway.Giveaway at-
                       tribute), 267
                                                                                                                                                                  tribute), 140
voter_count (aiogram.types.poll_option.PollOption at-
                                                                                                                                          winner_count (aiogram.types.giveaway completed.GiveawayCompleted
                       tribute), 267
                                                                                                                                                                  attribute), 141
                                                                                                                                          {\tt winner\_count}~(a iogram.types.give away\_winners.Give awayWinners
W
                                                                                                                                                                  attribute), 142
attribute), 142
                       attribute), 142
watcher(aiogram.types.proximity_alert_triggered.ProximityAnnyerneselection_date
                                                                                                                                                                  (aiogram.types.giveaway.Giveaway attribute),
                       attribute), 267
WEB_APP (aiogram.enums.menu_button_type.MenuButtonType
                                                                                                                                           winners_selection_date
                      attribute), 557
\verb|web_app|| (a iogram.types.in line\_keyboard\_button. In line KeyboardButto a iogram.types. give away\_winners. Give awayWinners and a line button and a lin
                                                                                                                                                                  attribute), 142
                       attribute), 183
web_app (aiogram.types.inline_query_results_button.Inline vite heresuals Battone (aiogram.types.transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transaction_partner_fragment.Transac
                                                                                                                                                                  attribute), 347
                      attribute), 325
```

```
wrap_local_file(aiogram.client.telegram.TelegramAPIServer
         attribute), 15
WRITE_ACCESS_ALLOWED
         (aiogram.enums.content_type.ContentType
         attribute), 550
write_access_allowed
         (aiogram.types.message.Message
                                             attribute),
         208
WriteAccessAllowed
                                   (class
                                                     in
         aiogram.types.write_access_allowed), 283
X
\verb|x_shift| (a iogram.types.mask_position.MaskPosition| at-
         tribute), 334
Υ
y\_shift (aiogram.types.mask_position.MaskPosition at-
         tribute), 334
{\tt year}~(a iogram. types. birth date. Birth date~attribute),~25
{\tt YELLOW}\,(aiogram.enums.topic\_icon\_color.TopicIconColor
         attribute), 561
YER (aiogram.enums.currency.Currency attribute), 554
Ζ
ZAR (aiogram.enums.currency.Currency attribute), 554
```