# aiogram Documentation

Release 3.7.0

aiogram Team

# **CONTENTS**

1 Features						
	1.1	Simple	usage			
	1.2	Usage v	without dispatcher			
2	Cont	ents	7			
	2.1	Installat	tion			
		2.1.1	From PyPI			
		2.1.2	From Arch Linux Repository			
		2.1.3	From PyPI			
		2.1.4	From GitHub			
	2.2	Migrati	on FAQ (2.x -> 3.0)			
-		2.2.1	Dispatcher			
		2.2.2	Filtering events			
		2.2.3	Bot API			
		2.2.4	Middlewares			
		2.2.5	Keyboard Markup			
		2.2.6	Callbacks data			
		2.2.7	Finite State machine			
		2.2.8	Sending Files			
		2.2.9	Webhook			
		2.2.10	Telegram API Server			
		2.2.11	Telegram objects transformation (to dict, to json, from json)			
	2.3	Bot AP	I			
		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?			
	2.4	Handlin	ng events			
		2.4.1	Router			
		2.4.2	Dispatcher			
		2.4.3	Dependency injection			
		2.4.4	Filtering events			
		2.4.5	Long-polling			
		2.4.6	Webhook			
		2.4.7	Finite State Machine			
		2.4.8	Middlewares			
		2.4.9	Errors			

	2.4.10	Flags
	2.4.11	Class based handlers
2.5	Utils .	
	2.5.1	Keyboard builder         562
	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
2.6	_	log
	2.6.1	3.7.0 [UNRELEASED DRAFT] (2024-05-10)
	2.6.2	3.6.0 (2024-05-06)
	2.6.3	3.5.0 (2024-04-23)
	2.6.4	3.4.1 (2024-02-17)
	2.6.5	3.4.0 (2024-02-16)
	2.6.6	3.3.0 (2023-12-31)
	2.6.7	3.2.0 (2023-11-24)
	2.6.8	3.1.1 (2023-09-25)
	2.6.9	3.1.0 (2023-09-22)
	2.6.10	3.0.0 (2023-09-01)
	2.6.11	3.0.0rc2 (2023-08-18)
	2.6.12	3.0.0rc1 (2023-08-06)
	2.6.13	3.0.0b9 (2023-07-30)
	2.6.14	3.0.0b8 (2023-07-17)
	2.6.15	3.0.067 (2023-02-18)
	2.6.16	3.0.0b6 (2022-11-18)
	2.6.17	3.0.0b5 (2022-10-02)
	2.6.18	3.0.0b4 (2022-08-14)
	2.6.19	3.0.0b3 (2022-04-19)
	2.6.20	3.0.0b2 (2022-02-19)
	2.6.21	3.0.0b1 (2021-12-12)
	2.6.22	3.0.0a18 (2021-11-10)
	2.6.23	3.0.0a17 (2021-09-24)
	2.6.24	3.0.0a16 (2021-09-22)
	2.6.25	3.0.0a15 (2021-09-10)
	2.6.27 2.6.28	2.14.3 (2021-07-21)
	2.6.29	2.14 (2021-07-27)
	2.6.30	2.13 (2021-04-28)
	2.6.31	2.12.1 (2021-04-28)
	2.6.32	2.12 (2021-03-22)
	2.6.33	2.11.2 (2021-03-14)
	2.6.34	2.11.1 (2021-11-10)
	2.6.35	2.11 (2021-11-10)
	2.6.36	2.10.1 (2021-11-08)
	2.6.37	2.10 (2021-09-14)
	2.6.38	2.9.2 (2021-06-13)
	2.6.39	2.9 (2021-06-08)
	2.6.40	2.8 (2021-04-26)
	2.6.41	2.7 (2021-04-07)
	2.6.42	2.6.1 (2021-01-25)
	2.0.72	2.0.1 (2021 01 23)

Index			631
Python 1	Module 1	Index	625
	2.7.5	Funding	. 624
		Take answers	
	2.7.3 2.7.4	Guides	
		Star on GitHub	
	2.7.1	Developing	
2.7	2.7.1	Daysloping	
2.7		( ,	
	2.6.76	0.1 (2017-06-03)	
	2.6.75	0.3.1 (2017-07-04)	
	2.6.74		
	2.6.72	0.3.2 (2017-07-04)	
	2.6.71	0.3.3 (2017-07-05)	
	2.6.70	0.4 (2017-08-05)	
	2.6.70	0.4.1 (2017-08-03)	
	2.6.68	1.0 (2017-11-19)	
	2.6.67	1.0.1 (2017-11-21)	
	2.6.66	1.0.2 (2017-11-29)	
	2.6.65	1.0.3 (2018-01-07)	
	2.6.64	1.0.4 (2018-01-10)	
	2.6.63	1.1 (2018-01-27)	
	2.6.62	1.2 (2018-02-23)	
	2.6.61	1.2.1 (2018-03-25)	
	2.6.60	1.2.2 (2018-04-08)	
	2.6.59	1.2.3 (2018-04-14)	
	2.6.58	1.3 (2021-04-22)	
	2.6.57	1.3.1 (2018-05-27)	
	2.6.56	1.3.2 (2021-05-27)	
	2.6.55	1.3.3 (2021-07-16)	. 618
	2.6.54	1.4 (2021-08-03)	. 618
	2.6.53	2.0 (2021-10-28)	. 618
	2.6.52	2.0.1 (2021-12-31)	. 618
	2.6.51	2.1 (2021-04-18)	
	2.6.50	2.2 (2021-06-09)	
	2.6.49	2.3 (2021-08-16)	
	2.6.48	2.4 (2021-10-29)	
	2.6.47	2.5 (2021-01-01)	
	2.6.46	2.5.1 (2021-01-01)	
	2.6.45	2.5.2 (2021-01-01)	
	2.6.44	2.5.3 (2021-01-25)	
	2.6.43	2.6 (2021-01-23)	615

**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 7.3 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.)
   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!")
```

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

(continues on next page)

(continued from previous page)

```
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

pacman -S python-aiogram

Development build (3.x)

# 2.1.3 From PyPI

pip install -U aiogram

# 2.1.4 From GitHub

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

# 2.2 Migration FAQ (2.x -> 3.0)

**Danger:** 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.

**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 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 (Read more »).
- With routes, you can easily modularize your code and potentially share these modules between projects.
- 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 <code>aiogram.methods.delete\_webhook.DeleteWebhook</code> method directly, rather than passing <code>skip\_updates=True</code> to the start polling method.

# 2.2.2 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.

# 2.2.3 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.

#### 2.2.4 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 »)

# 2.2.5 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 »)

# 2.2.6 Callbacks data

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

# 2.2.7 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.

# 2.2.8 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* »)

#### 2.2.9 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).

# 2.2.10 Telegram API Server

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

# 2.2.11 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 TelegramObject.model\_dump\_json() (Read more)
- TelegramObject.to\_python() should be replaced by TelegramObject.model\_dump() (Read more)

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

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

# Version 3.x
    message_json = message.model_dump_json()
    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'>

# Version 3.x
    message_dict = message.model_dump()
    print(message_dict)
    # {"id": 42, ...}
    print(type(message_dict))
# { class 'dict'>
```

# 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.

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, parse\_mode: str | None = None, disable\_web\_page\_preview: bool | None = None, protect\_content: bool | None = None, default: DefaultBotProperties | None = None) → 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.
- parse\_mode Default parse mode. If specified it will be propagated into the API methods at runtime.
- **disable\_web\_page\_preview** Default disable\_web\_page\_preview mode. If specified it will be propagated into the API methods at runtime.
- **protect\_content** Default protect\_content mode. If specified it will be propagated into the API methods at runtime.
- 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) <math>\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:

```
session = AiohttpSession(
    api=TelegramAPIServer.from_base('http://localhost:8082')
)
bot = Bot(..., session=session)
```

```
 \textbf{class} \  \, \textbf{aiogram.client.telegram.TelegramAPIServer} (\textit{base: str, file: str, is\_local: bool = False,} \\ \textit{wrap\_local\_file:} \\ \textit{-aiogram.client.telegram.FilesPathWrapper} = \\ \textit{-aiogram.client.telegram.BareFilesPathWrapper} \\ \textit{object>})
```

Base config for API Endpoints

```
api_url(token: str, method: str) → str
Generate URL for API methods
```

#### **Parameters**

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

# Returns

URL

```
base: str
           Base URL
     file: str
           Files URL
     file_url(token: str, path: str) \rightarrow str
           Generate URL for downloading files
               Parameters
                   • token - Bot token
                   • path - file path
               Returns
                   URL
     classmethod from_base(base: str, **kwargs: Any) → 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[[...],
                                                         ~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.
     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 \mid None = None) \rightarrow TelegramType
```

Make request to Telegram Bot API

#### **Parameters**

- bot Bot instance
- **method** Method instance
- timeout Request timeout

#### Returns

#### Raises

TelegramApiError -

```
prepare_value(value: Any, bot: Bot, files: Dict[str, Any], _dumps_json: bool = True) \rightarrow Any Prepare value before send
```

```
abstract async stream_content(url: str, headers: Dict[str, Any] \mid None = None, timeout: int = 30, chunk_size: int = 65536, raise_for_status: bool = True) <math>\rightarrow AsyncGenerator[bytes, None]
```

Stream reader

#### aiohttp

AiohttpSession represents a wrapper-class around ClientSession from aiohttp

Currently AiohttpSession is a default session used in aiogram.Bot

```
class aiogram.client.session.aiohttp.AiohttpSession(proxy: Iterable[str \mid Tuple[str, BasicAuth]] \mid str \mid Tuple[str, BasicAuth] \mid None = None, **kwargs: Any)
```

#### 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)
```

**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")
```

**Note:** 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**

2.3. Bot API

#### 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
        self.ignore_methods = ignore_methods if ignore_methods else []

async def __call__(
        self,
        make_request: NextRequestMiddlewareType[TelegramType],
```

(continues on next page)

17

(continued from previous page)

```
bot: "Bot",
13
           method: TelegramMethod[TelegramType],
       ) -> Response[TelegramType]:
15
           if type(method) not in self.ignore_methods:
                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)
```

**Note:** 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 sender

#### height: int

Video height as defined by sender

#### duration: int

Duration of the video in seconds as defined by sender

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

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

```
model_post_init(\_ModelMetaclass\_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 sender

```
file_name: str | None
```

Optional. Original animation filename as defined by sender

```
mime_type: str | None
```

Optional. MIME type of the file as defined by 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.

# **Audio**

```
class aiogram.types.audio.Audio(*, file_id: str, file_unique_id: str, duration: int, performer: str | None = None, title: str | None = None, file_name: str | None = None, mime_type: str | None = None, file_size: int | None = None, thumbnail: PhotoSize | None = None, **extra_data: Any)
```

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 sender

#### performer: str | None

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

```
title: str | None
```

Optional. Title of the audio as defined by 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(\_ModelMetaclass\_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 sender

```
mime_type: str | None
```

Optional. MIME type of the file as defined by 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.

```
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
- 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(\_ModelMetaclass\_\_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],

\*\*extra 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(\_ModelMetaclass\_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(\_ModelMetaclass\_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(\_ModelMetaclass\_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(\_ModelMetaclass\_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(\_ModelMetaclass\_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#backgroundtypefill

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

Type of the background, always 'fill'

fill: BackgroundFillSolid | BackgroundFillGradient | 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(\_ModelMetaclass\_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

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(\_ModelMetaclass\_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 | 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(_ModelMetaclass__context: Any) → 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.

 $model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None$ 

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.

 $model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None$ 

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:

- 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(\_ModelMetaclass\_context: Any) \rightarrow None
```

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

# **BotCommandScopeAllChatAdministrators**

```
{\bf class}\ {\bf aiogram.types.bot\_command\_scope\_all\_chat\_administrators.} {\bf BotCommandScopeAllChatAdministrators} (*, {\bf class}\ {\bf class}\ {\bf class}\ {\bf class}\ {\bf class}\ {\bf class}\ {\bf copeAllChatAdministrators} (*, {\bf class}\ {
```

type: Literal[i = Bot-Command Scope

> \*\*extra\_d 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(\_ModelMetaclass\_context: Any) \rightarrow None
```

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

#### **BotCommandScopeAllGroupChats**

```
{\bf class}\ aiogram.types.bot\_command\_scope\_all\_group\_chats. {\bf BotCommandScopeAllGroupChats}(*, {\it type:} {\bf class}) aiogram.types.bot\_command\_scope\_all\_group\_chats. (*, {\it type:} {\bf class}) aio
```

eral[BotCommandScopeTy = Bot-Com-

mand-

ScopeType.ALL\_GROUP\_ \*\*ex-

\*\*extra\_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(\_ModelMetaclass\_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:
Literal[BotCommandScoor=
BotCommand-

ScopeType.ALL\_PRI

\*\*ex-

\*\*extra\_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(\_ModelMetaclass\_context: Any) \rightarrow None
```

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

# BotCommandScopeChat

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(\_ModelMetaclass\_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(\_ModelMetaclass\_\_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(\_ModelMetaclass\_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(\_ModelMetaclass\_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(\_ModelMetaclass\_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(_ModelMetaclass__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(ModelMetaclass 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(\_ModelMetaclass\_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(\_ModelMetaclass\_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.

 $model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None$ 

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.

 $model_post_init(\_ModelMetaclass\_\_context: Any) \rightarrow None$ 

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.

 $model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None$ 

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(\_ModelMetaclass\_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**

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 <code>message</code> will be present. If the button was attached to a message sent via the bot (in inline mode), the field <code>inline\_message\_id</code> will be present. Exactly one of the fields <code>data</code> or <code>game\_short\_name</code> will be present.

**NOTE:** After the user presses a callback button, Telegram clients will display a progress bar until you call <code>aiogram.methods.answer\_callback\_query.AnswerCallbackQuery</code>. It is, therefore, necessary to react by calling <code>aiogram.methods.answer\_callback\_query.AnswerCallbackQuery</code> 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(\_ModelMetaclass\_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

```
instance of method aiogram.methods.answer_callback_query.
AnswerCallbackQuery
```

### 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] | 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 | None = None, description: str |
                                     None = None, emoji\_status\_custom\_emoji\_id: str | None = None,
                                     emoji status expiration date: datetime | 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 \mid 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 \mid 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 | 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] | 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 aiogram.methods.get\_chat.GetChat.

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.

# $model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None$

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 \*  $10^{\circ}$  (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  $aiogram.methods.unban\_chat\_sender\_chat.UnbanChatSenderChat$  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.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

instance of method aiogram.methods.edit\_chat\_invite\_link.
EditChatInviteLink

 $create_invite_link(name: str | None = None, expire_date: datetime.datetime | datetime.timedelta | int | None = None, member_limit: int | None = None, creates_join_request: bool | None = None, **kwargs: Any) <math>\rightarrow CreateChatInviteLink$ 

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 aiogram.methods.send\_chat\_action.SendChatAction 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 <code>aiogram.methods.get\_chat\_member.GetChatMember</code> 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 <code>aiogram.methods.get\_chat\_member\_count.GetChatMemberCount</code> 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

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

 $unpin_message(message\_id: int \mid None = None, **kwargs: Any) \rightarrow UnpinChatMessage$ 

Shortcut for method <code>aiogram.methods.unpin\_chat\_message.UnpinChatMessage</code> 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**

**message\_id** – Identifier of a message to unpin. If not specified, the most recent pinned message (by sending date) will be unpinned.

#### Returns

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

 $pin_message(message\_id: int, disable\_notification: bool | None = None, **kwargs: Any) \rightarrow PinChatMessage$ 

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
- **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

```
\textbf{set\_administrator\_custom\_title}(\textit{user\_id}: \textit{int}, \textit{custom\_title}: \textit{str}, **kwargs: Any}) \rightarrow \textit{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

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 aiogram.methods.set\_chat\_permissions.SetChatPermissions 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 <code>aiogram.methods.promote\_chat\_member.PromoteChatMember</code> 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

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

**restrict**( $user\_id: int, permissions: ChatPermissions, use\_independent\_chat\_permissions: bool | None = None, until\_date: datetime.datetime | datetime.timedelta | int | None = None, **kwargs: Any) <math>\rightarrow$  RestrictChatMember

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

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 <code>aiogram.methods.unban\_chat\_member.UnbanChatMember</code> 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 | datetime.timedelta | <math>int | None = None,  $revoke\_messages$ : bool | 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.

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

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

Shortcut for method <code>aiogram.methods.set\_chat\_description.SetChatDescription</code> 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

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

```
\textbf{set\_photo}(\textit{photo}: InputFile, **kwargs: Any) \rightarrow \textit{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} unpin\_all\_general\_forum\_topic\_messages(**kwargs: Any) $\rightarrow$ 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\_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:  $bool \mid None = None,$ 

can\_manage\_chat: bool,

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

\*\*extra\_data: Any)

## 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.

# $model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None$

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

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(\_ModelMetaclass\_context: Any) \rightarrow None
```

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

# **ChatBoost**

52

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(\_ModelMetaclass\_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

```
\textbf{class} \  \, \textbf{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.

 $model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None$ 

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

## ChatBoostRemoved

**class** aiogram.types.chat\_boost\_removed.**ChatBoostRemoved**(\*, chat: Chat, boost\_id: str, remove\_date:

datetime, source:

ChatBoostSourcePremium |

ChatBoostSourceGiftCode |

 $Chat Boost Source Give away, **extra\_data:$ 

Any)

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.

 $model_post_init(ModelMetaclass context: Any) \rightarrow None$ 

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(ModelMetaclass context: Any) \rightarrow None
```

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

### ChatBoostSourceGiftCode

user: User,

\*\*extra\_data: Any)

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(ModelMetaclass 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

```
 \textbf{class} \  \, \textbf{aiogram.types.chat\_boost\_source\_giveaway.ChatBoostSourceGiveaway}(*, source: Literal[ChatBoostSourceType.GIVEAWAY] \\ = ChatBoostSource-\\ Type.GIVEAWAY, \\ giveaway\_message\_id: \\ int, user: \  \, \textbf{User} \mid None = \\ None, is\_unclaimed: \\ bool \mid None = None, \\ \end{cases}
```

\*\*extra\_data: Any)

The boost was obtained by the creation of a Telegram Premium giveaway. This boosts the chat 4 times for the duration of the corresponding Telegram Premium subscription.

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.

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

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

```
model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None
```

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

```
user: User | None
```

Optional. User that won the prize in the giveaway if any

```
is_unclaimed: bool | None
```

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

# **ChatBoostSourcePremium**

Type.PREMIUM, user: User, \*\*extra\_data: Any)

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(\_ModelMetaclass\_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.

 $model_post_init(\_ModelMetaclass\_\_context: Any) \rightarrow None$ 

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* | *None* = *None*, *first\_name: str* | *None* = None, last name:  $str \mid None = None$ , is forum: bool | None = None, accent\_color\_id: int, active usernames:  $List[str] \mid None = None$ , available reactions: List[ReactionTypeEmoji | ReactionTypeCustomEmoji $l \mid None = None$ , 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 | None = *None*, description: str | None = None,emoji\_status\_custom\_emoji\_id: str | None = None, emoji\_status\_expiration\_date: datetime | None = None, has aggressive anti spam enabled: bool None = None, has\_hidden\_members: bool | None = None, has private forwards: bool | None = None,  $has\_protected\_content: bool \mid None = None,$ has restricted voice and video messages: bool | None = None, has\_visible\_history: bool | None = *None*, *invite* link:  $str \mid None = None$ , *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 |  $None = None, slow\_mode\_delay: int | None = None,$  $sticker\ set\ name:\ str\mid None=None,$  $unrestrict\_boost\_count$ :  $int \mid None = None$ , max\_reaction\_count: int, \*\*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] | 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.

# $model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None$

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 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

### 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

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

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

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

```
model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None
```

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

name: str | None

Optional. Invite link name

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

# **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 <code>aiogram.methods.approve\_chat\_join\_request.ApproveChatJoinRequest</code> 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

instance of method aiogram.methods.approve\_chat\_join\_request.
ApproveChatJoinRequest

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

Shortcut for method <code>aiogram.methods.decline\_chat\_join\_request.DeclineChatJoinRequest</code> 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')>, 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

- **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_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')>, 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
- 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, has\_spoiler: Optional[bool] = None, disable\_notification: Optional[bool] = None, protect\_content: Optional[Union[bool, Default]] = <Default('protect\_content')>, reply\_parameters: Optional[Union[bool, Default]] = <Default('protect\_content')>, 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) → 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*
- 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
- 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

```
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, has_spoiler: Optional[bool] = None, 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) → 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*
- 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
- **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 <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
- **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

- **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')>, 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 aiogram.methods.send\_contact.SendContact 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
- 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\_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
- 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

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
- 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

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
- 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

**answer\_game** ( $game\_short\_name$ : str,  $business\_connection\_id$ : Optional[str] = None,  $message\_thread\_id$ : Optional[int] = None,  $disable\_notification$ : Optional[bool] = None,  $protect\_content$ :  $Optional[Union[bool, Default]] = <Default('protect\_content')>$ ,  $reply\_parameters$ : Optional[ReplyParameters] = None,  $reply\_markup$ : Optional[InlineKeyboardMarkup] = None,  $allow\_sending\_without\_reply$ : Optional[bool] = None,  $reply\_to\_message\_id$ : Optional[int] = None, \*\*kwargs: Any)  $\rightarrow SendGame$ 

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.

### **Parameters**

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

- 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
- 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 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
- **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

answer\_invoice(title: str, description: str, payload: str, provider\_token: str, currency: str, prices:

List[LabeledPrice], message\_thread\_id: Optional[int] = 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') >, 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 for your internal processes.
- provider\_token Payment provider token, obtained via @BotFather
- **currency** Three-letter ISO 4217 currency code, see more on currencies
- **prices** Price breakdown, a JSON-serialized list of components (e.g. product price, tax, discount, delivery cost, delivery tax, bonus, etc.)
- **message\_thread\_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- 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
- **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
- **need\_phone\_number** Pass True if you require the user's phone number to complete the order
- need\_email Pass True if you require the user's email address to complete the order

- need\_shipping\_address Pass True if you require the user's shipping address to complete the order
- **send\_phone\_number\_to\_provider** Pass True if the user's phone number should be sent to provider
- **send\_email\_to\_provider** Pass True if the user's email address should be sent to provider
- is\_flexible Pass True if the final price depends on the shipping method
- **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** 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, provider\_token: str, currency: str, prices:

List[LabeledPrice], message\_thread\_id: Optional[int] = 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')>, 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 for your internal processes.
- **provider\_token** Payment provider token, obtained via @BotFather

- **currency** Three-letter ISO 4217 currency code, see more on currencies
- **prices** Price breakdown, a JSON-serialized list of components (e.g. product price, tax, discount, delivery cost, delivery tax, bonus, etc.)
- message\_thread\_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- 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
- **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
- need\_phone\_number Pass True if you require the user's phone number to complete the order
- need\_email Pass True if you require the user's email address to complete the order
- **need\_shipping\_address** Pass True if you require the user's shipping address to complete the order
- **send\_phone\_number\_to\_provider** Pass True if the user's phone number should be sent to provider
- **send\_email\_to\_provider** Pass True if the user's email address should be sent to provider
- is\_flexible Pass True if the final price depends on the shipping method
- 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** 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</code>. SendLocation 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
- **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

answer\_location\_pm(latitude: float, longitude: float, business\_connection\_id: Optional[str] = None, message\_thread\_id: Optional[int] = None, 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')>, 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 <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

- **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

answer\_media\_group(media: List[Union[InputMediaAudio, InputMediaDocument, InputMediaPhoto, InputMediaVideo]], 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')>, reply\_parameters: Optional[ReplyParameters] = None, allow\_sending\_without\_reply: Optional[bool] = None, reply\_to\_message\_id: Optional[int] = None, \*\*kwargs: Any) → 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
- 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

 $instance\ of\ method\ aiogram. \textit{methods.send\_media\_group.} Send \texttt{MediaGroup}$ 

```
\begin{tabular}{ll} \textbf{answer_media\_group\_pm} (media: List[Union[InputMediaAudio, InputMediaDocument, InputMediaPhoto, InputMediaVideo]], 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') >, reply\_parameters: Optional[ReplyParameters] = \\ None, allow\_sending\_without\_reply: Optional[bool] = None, \\ reply\_to\_message\_id: Optional[int] = None, **kwargs: Any) \rightarrow \\ SendMediaGroup \end{tabular}
```

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
- 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*
- 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
- 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\_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*
- 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
- **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\_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')>, reply_parameters:

Optional[ReplyParameters] = None, reply_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, 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
- **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')>,

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
- **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 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
- 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

```
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')>, reply_parameters: Optional[ReplyParameters] = None, reply_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardMemove, 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
- 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
- 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 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

### **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
- 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

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')>, 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
- 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

```
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')>, 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
- **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

Shortcut for method <code>aiogram.methods.send\_video.SendVideo</code> 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 <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* »
- 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*
- 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
- 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

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

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

```
model_post_init(\_ModelMetaclass\_\_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*
- 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
- **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 <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
- 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\_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
- 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</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
- **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 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
- 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(\_ModelMetaclass\_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(\_ModelMetaclass\_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\_aetete\_storie. !- - -!

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.

## $model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None$

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, \*\*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.

 $model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None$ 

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.

 $model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None$ 

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

user: *User* 

Information about the user

## ChatMemberMember

class aiogram.types.chat\_member\_member.ChatMemberMember(\*, status:

Literal[ChatMemberStatus.MEMBER] =
ChatMemberStatus.MEMBER, user: User,
\*\*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'

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

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

 $model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None$ 

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

user: User

Information about the user

### **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.

 $model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None$ 

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: Lit-

eral[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, \*\*extra\_data: Any)

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

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

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

## $model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None$

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, old\_chat\_member:

ChatMemberOwner |

ChatMemberAdministrator |

ChatMemberMember |

ChatMemberRestricted |

ChatMemberLeft | ChatMemberBanned,

new\_chat\_member: ChatMemberOwner

| ChatMemberAdministrator |

ChatMemberMember |

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 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')>, 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
- 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, has\_spoiler: Optional[bool] = None, disable\_notification: Optional[bool] = None, protect\_content: Optional[Union[bool, Default]] = < Default('protect\_content') >, reply\_parameters: Optional[Union[bool, Default]] = < Default('protect\_content') >, 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) → 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*
- 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
- 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 <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
- 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 <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
- 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

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
- 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\_document.SendDocument

**answer\_game**( $game\_short\_name$ : str,  $business\_connection\_id$ : Optional[str] = None,  $message\_thread\_id$ : Optional[int] = None,  $disable\_notification$ : Optional[bool] = None,  $protect\_content$ :  $Optional[Union[bool, Default]] = <Default('protect\_content')>$ ,  $reply\_parameters$ : Optional[ReplyParameters] = None,  $reply\_markup$ : Optional[InlineKeyboardMarkup] = None,  $allow\_sending\_without\_reply$ : Optional[bool] = None,  $reply\_to\_message\_id$ : Optional[int] = None, \*\*kwargs: Any)  $\rightarrow SendGame$ 

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
- 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

answer\_invoice(title: str, description: str, payload: str, provider\_token: str, currency: str, prices:

List[LabeledPrice], message\_thread\_id: Optional[int] = 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') >, 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 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

## **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 for your internal processes.
- provider\_token Payment provider token, obtained via @BotFather
- **currency** Three-letter ISO 4217 currency code, see more on currencies
- **prices** Price breakdown, a JSON-serialized list of components (e.g. product price, tax, discount, delivery cost, delivery tax, bonus, etc.)
- message\_thread\_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- 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
- **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
- need\_phone\_number Pass True if you require the user's phone number to complete the order
- need\_email Pass True if you require the user's email address to complete the order
- **need\_shipping\_address** Pass True if you require the user's shipping address to complete the order
- **send\_phone\_number\_to\_provider** Pass True if the user's phone number should be sent to provider
- **send\_email\_to\_provider** Pass True if the user's email address should be sent to provider
- is\_flexible Pass True if the final price depends on the shipping method

- **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** 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</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
- 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  $aiogram.methods.send\_media\_group.SendMediaGroup$  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
- 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*
- 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
- 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\_photo.SendPhoto

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

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

```
model_post_init(\_ModelMetaclass\_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[Union[datetime.datetime, datetime.timedelta, int]] = None, protect\_content:

Optional[Union[bool] = None, 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) → 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
- **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
- 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 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

#### **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
- **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')>, 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
- 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

Shortcut for method <code>aiogram.methods.send\_video.SendVideo</code> 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*
- 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
- 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
- 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 <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
- 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\_voice.SendVoice

### **ChatPermissions**

 $\textbf{class} \ \, \textbf{aiogram.types.chat\_permissions.ChatPermissions} (*, \textit{can\_send\_messages: bool} \mid \textit{None} = \textit{None}, \\$ 

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\_change\_info: 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.

 $model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None$ 

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(\_ModelMetaclass\_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(\_ModelMetaclass\_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**

```
class aiogram.types.contact.Contact(*, phone_number: str, first_name: str, last_name: str | None = None,
                                            user_id: int | None = None, vcard: str | None = None, **extra_data:
                                            Any)
     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 ComputedFieldInfo objects.
     model_post_init(\_ModelMetaclass\_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(\_ModelMetaclass\_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 sender

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

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

```
model_post_init(\_ModelMetaclass\_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 sender

```
mime_type: str | None
```

Optional. MIME type of the file as defined by 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, 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 super-
           group 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
```

2.3. Bot API 129

Optional. Message is an audio file, information about the file

Optional. Message is a general file, information about the file

Optional. Message is a photo, available sizes of the photo

document: Document | None

photo: List[PhotoSize] | None

130

```
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
model_computed_fields: ClassVar[dict[str, ComputedFieldInfo]] = {}
     A dictionary of computed field names and their corresponding ComputedFieldInfo objects.
model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None
     We need to both initialize private attributes and call the user-defined model_post_init method.
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**

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(\_ModelMetaclass\_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**

```
class aiogram.types.force_reply.ForceReply(*, force\_reply: Literal[True] = True, input\_field\_placeholder: str \mid None = None, selective: bool \mid None = None, **extra_data: Any)
```

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</code>. ForceReply in your bot's questions, it will receive the user's answers even if it only receives replies, commands and mentions - without any extra work for the user.

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(\_ModelMetaclass\_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(\_ModelMetaclass\_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(\_ModelMetaclass\_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.

 $model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None$ 

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.

 $model\_post\_init(\_\mathit{ModelMetaclass}\_\mathit{context} : \mathit{Any}) \rightarrow None$ 

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(\_ModelMetaclass\_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(\_ModelMetaclass\_\_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(\_ModelMetaclass\_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

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

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

 ${\bf model\_post\_init}(\_{\it ModelMetaclass\_context}: {\it Any}) \rightarrow {\it None}$ 

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

has\_public\_winners: bool | None

Optional. True, if the list of giveaway winners will be visible to everyone

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.

premium\_subscription\_month\_count: int | None

*Optional*. The number of months the Telegram Premium subscription won from the giveaway will be active for

## GiveawayCompleted

class aiogram.types.giveaway\_completed.GiveawayCompleted(\*, winner\_count: int,

unclaimed\_prize\_count: int | None = None, giveaway\_message: Message | 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.

 $model_post_init(\_ModelMetaclass\_\_context: Any) \rightarrow None$ 

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

## **GiveawayCreated**

```
class aiogram.types.giveaway_created.GiveawayCreated(**extra_data: Any)
```

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

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

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

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

```
model_post_init(\_ModelMetaclass\_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, winner_count: int, winners: List[User], additional_chat_count: int | None = None, premium_subscription_month_count: int | None = None, unclaimed_prize_count: int | None = None, only_new_members: bool | None = None, was_refunded: bool | None = None, prize_description: str | None = None, **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.

 $model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None$ 

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

# premium\_subscription\_month\_count: int | None

Optional. The number of months the Telegram Premium subscription won from the giveaway will be active for

## 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

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

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

```
model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None
```

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

## date: Literal[0]

Always 0. The field can be used to differentiate regular and inaccessible messages.

### 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. You **must** use exactly one of the optional fields.

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 button is pressed, 1-64 bytes. Not supported for messages sent on behalf of a Telegram Business account.

## 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 aiogram.methods. answer\_web\_app\_query. AnswerWebAppQuery. 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.

## $model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None$

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.

# InlineKeyboardMarkup

```
class aiogram.types.inline_keyboard_markup.InlineKeyboardMarkup(*, inline_keyboard:
                                                                               List/List/InlineKeyboardButton]],
                                                                               **extra_data: Any)
     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(ModelMetaclass 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]
class aiogram.types.input_file.FSInputFile(path: str | Path, filename: str | None = None, chunk_size: int
                                                     =65536)
     async read(bot: Bot) \rightarrow AsyncGenerator[bytes, None]
class aiogram.types.input_file.URLInputFile(url: str, headers: Dict[str, Any] | None = None, filename:
                                                      str \mid None = None, chunk\_size: int = 65536, timeout: int =
                                                      30, bot: 'Bot' \mid None = None)
```

2.3. Bot API 139

**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(\_ModelMetaclass\_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-
```

~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: ~typing.List[~aiogram.types.message\_entity.MessageEntit]
| None = None, width: int | None =
None, height: int | None = None,
duration: int | None = None,

ing.Literal[InputMediaType.ANIMATION]

= InputMediaType.ANIMATION,

has\_spoiler: bool | None = None, \*\*extra data: ~typing.Any)

media: str |

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.

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

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

 $model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None$ 

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

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:
~typing.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(\_ModelMetaclass\_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

~typing.Any)

```
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:
```

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(ModelMetaclass 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, 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
     model_computed_fields: ClassVar[dict[str, ComputedFieldInfo]] = {}
           A dictionary of computed field names and their corresponding ComputedFieldInfo objects.
     model_post_init(\_ModelMetaclass\_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
```

Optional. Pass True if the photo needs to be covered with a spoiler animation

## InputMediaVideo

has\_spoiler: bool | None

```
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:
                                                                   ~typ-
                                                                   ing.List[~aiogram.types.message_entity.MessageEntity]
                                                                   | None = None, width: int | None = None,
                                                                   height: int \mid None = None, duration: int \mid
                                                                   None = None, supports_streaming: bool |
                                                                   None = None, has_spoiler: bool | None =
                                                                   None, **extra_data: ~typing.Any)
     Represents a video to be sent.
     Source: https://core.telegram.org/bots/api#inputmediavideo
     type: Literal[InputMediaType.VIDE0]
           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 »
     thumbnail: InputFile | 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 »
     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(\_ModelMetaclass\_context: Any) \rightarrow None
           We need to both initialize private attributes and call the user-defined model_post_init method.
     width: int | None
```

2.3. Bot API 145

Optional. Video width

Optional. Video height

height: int | None

```
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

# InputPollOption

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

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(\_ModelMetaclass\_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. For simple text buttons, *String* can be used instead of this object to specify the button text. The optional fields *web\_app*, *request\_users*, *request\_chat*, *request\_contact*, *request\_location*, and *request\_poll* are mutually exclusive. **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(\_ModelMetaclass\_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

# 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(\_ModelMetaclass\_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,
```

bool. chat\_is\_forum:  $bool \mid None =$ None, chat\_has\_username: bool | None = None, chat is created: bool | None = None, user\_administrator\_rights: ChatAdministratorRights | None = None. bot\_administrator\_rights: ChatAdministra-

chat\_is\_channel:

None, bot\_is\_member:

torRights | None =

bool | None =

None,

request\_title: bool | None = None, request\_username: bool | None = None,

request\_photo: bool | None = None,

\*\*extra\_data:
Anv)

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(\_ModelMetaclass\_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

```
class aiogram.types.keyboard_button_request_user.KeyboardButtonRequestUser(*, request_id: int,
```

```
user_is_bot: bool |
None = None,
user_is_premium:
bool | None =
None,
**extra_data:
Any)
```

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(\_ModelMetaclass\_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

class aiogram.types.keyboard\_button\_request\_users.KeyboardButtonRequestUsers(\*, request\_id:

int. user is bot: bool | None = None. user\_is\_premium: bool | None = None, max\_quantity: *int* | *None* = None. request\_name:  $bool \mid None =$ 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 ComputedFieldInfo objects.

```
model_post_init(\_ModelMetaclass\_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

# **LinkPreviewOptions**

```
class aiogram.types.link_preview_options.LinkPreviewOptions(*, is_disabled: bool |
```

~aiogram.client.default.Default | None = < Default('link\_preview\_is\_disabled')>,  $url: str \mid None = None,$ prefer\_small\_media: bool | ~aiogram.client.default.Default | None = <Default('link\_preview\_prefer\_small\_media')>, prefer\_large\_media: bool | ~aiogram.client.default.Default | None = < Default('link\_preview\_prefer\_large\_media')>, show above text: bool | ~aiogram.client.default.Default | None = <Default('link\_preview\_show\_above\_text')>, \*\*extra\_data: ~typing.Any)

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(\_ModelMetaclass\_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

This object represents a point on the map.

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

latitude: float

Latitude as defined by sender

longitude: float

Longitude as defined by 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(\_ModelMetaclass\_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(\_ModelMetaclass\_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(\_ModelMetaclass\_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 aiogram.types.menu\_button\_default.MenuButtonDefault is set for a private chat, then it is applied in the chat. Otherwise the default menu button is applied. By default, the menu button opens the list of bot commands.

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(\_ModelMetaclass\_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 aiogram.methods. answer\_web\_app\_query.AnswerWebAppQuery.

#### **MenuButtonCommands**

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(\_ModelMetaclass\_\_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)

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.

 $model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None$ 

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.

 $model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None$ 

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 aiogram.methods. answer\_web\_app\_query.AnswerWebAppQuery.

Message

class aiogram.types.message.Message(\*, message id: int, date: datetime, 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,  $author\_signature$ :  $str \mid None = None$ , text:  $str \mid None = None$ *None*, *entities*: *List*[MessageEntity] | *None* = *None*, *link\_preview\_options*: LinkPreviewOptions | *None* = *None*, animation: Animation | None = None, audio: Audio | None = None, document: Document | None = None, photo: List[PhotoSize] | None = None, sticker: Sticker | None = None, story: Story | None = None, *video*: Video | *None* = *None*, *video note*: VideoNote | *None* = *None*, voice: Voice | None = None, caption: str | None = None, *caption\_entities: List*[MessageEntity] | *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,  $supergroup\_chat\_created$ :  $bool \mid None = None$ ,  $channel\_chat\_created: bool | None = None,$ 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, 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:* GiveawayWinners | *None* = *None*, *giveaway\_completed*: GiveawayCompleted | *None* = *None*, *video\_chat\_scheduled*: VideoChatScheduled | None = None, video chat started: VideoChatStarted | None = None, video\_chat\_ended: VideoChatEnded | None = None, video\_chat\_participants\_invited: VideoChatParticipantsInvited | None = None, web\_app\_data:

WebAppData | None = None, reply markup: InlineKeyboardMarkup

forward from: User | None = None, forward from chat: Chat | None

| None = None, forward date: datetime | None = None,

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

#### sender\_chat: Chat | None

*Optional*. Sender of the message, sent on behalf of a chat. For example, the channel itself for channel posts, the supergroup itself for messages from anonymous group administrators, the linked channel for messages automatically forwarded to the discussion group. For backward compatibility, the field *from* contains a fake sender user in non-channel chats, if the message was sent on behalf of a chat.

#### sender\_boost\_count: int | 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
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
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
```

2.3. Bot API 159

Optional. Message is a video, information about the video

video: Video | None

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, 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 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 »

#### 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
general_forum_topic_unhidden: GeneralForumTopicUnhidden | None
    Optional. Service message: the 'General' forum topic unhidden
model_computed_fields: ClassVar[dict[str, ComputedFieldInfo]] = {}
    A dictionary of computed field names and their corresponding ComputedFieldInfo objects.
model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None
    We need to both initialize private attributes and call the user-defined model_post_init method.
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 administrator, 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
```

```
reply_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,
has_spoiler: Optional[bool] = None, 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, **kwargs:
Any) → SendAnimation
```

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

- chat\_id
- message\_thread\_id
- business\_connection\_id
- reply\_to\_message\_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 »
- 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*
- 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
- 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

#### Returns

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, has_spoiler: Optional[bool] = None, 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, 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

#### **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*
- 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

- **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')>, reply\_parameters: Optional[ReplyParameters] = 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\_to\_message\_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
- 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

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')>, 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
- 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

```
 \begin{split} \textbf{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')>, reply\_parameters: \\ &Optional[ReplyParameters] = None, reply\_markup: \\ &Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, \\ &ForceReply]] = None, allow\_sending\_without\_reply: Optional[bool] = None, **kwargs: \\ &Any) \rightarrow SendContact \end{split}
```

Shortcut for method aiogram.methods.send\_contact.SendContact will automatically fill method attributes:

- chat id
- message\_thread\_id
- business\_connection\_id
- reply\_to\_message\_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
- **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

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

```
\begin{tabular}{ll} \textbf{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]] = & Optional[votect\_content') & 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 \\ \end{tabular}
```

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
- 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')>, reply\_parameters: Optional[ReplyParameters] = None,
reply\_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup,
ReplyKeyboardRemove, ForceReply]] = None, allow\_sending\_without\_reply:
Optional[bool] = 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
- reply\_to\_message\_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

- **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

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')>, 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
- 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

 $instance\ of\ method\ aiogram.methods.send\_document.SendDocument$ 

 $\textbf{reply\_game} (game\_short\_name: str, disable\_notification: Optional[bool] = None, protect\_content: \\ Optional[Union[bool, Default]] = <Default('protect\_content')>, reply\_parameters: \\ Optional[ReplyParameters] = None, reply\_markup: Optional[InlineKeyboardMarkup] = None, \\ allow\_sending\_without\_reply: Optional[bool] = None, **kwargs: Any) \rightarrow SendGame$ 

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

- chat\_id
- message\_thread\_id
- business\_connection\_id
- reply\_to\_message\_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
- 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

#### Returns

instance of method aiogram.methods.send\_game.SendGame

**answer\_game**( $game\_short\_name$ : str,  $disable\_notification$ : Optional[bool] = None,  $protect\_content$ :  $Optional[Union[bool, Default]] = <Default('protect\_content')>$ ,  $reply\_parameters$ : Optional[ReplyParameters] = None,  $reply\_markup$ : Optional[InlineKeyboardMarkup] = None,  $allow\_sending\_without\_reply$ : Optional[bool] = None,  $reply\_to\_message\_id$ : Optional[int] = None, \*\*kwargs: Any)  $\rightarrow SendGame$ 

Shortcut for method aiogram.methods.send\_game.SendGame 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
- 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

```
reply_invoice(title: str, description: str, payload: str, provider_token: str, currency: str, prices:

List[LabeledPrice], max_tip_amount: Optional[int] = None, suggested_tip_amounts:

Optional[List[int]] = None, 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_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')>, reply_parameters: Optional[ReplyParameters] = None, reply_markup: Optional[InlineKeyboardMarkup] = None, allow_sending_without_reply:
Optional[bool] = None, **kwargs: Any) → SendInvoice
```

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

- chat\_id
- message\_thread\_id
- business\_connection\_id
- reply\_to\_message\_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 for your internal processes.
- **provider\_token** Payment provider token, obtained via @BotFather
- **currency** Three-letter ISO 4217 currency code, see more on currencies
- **prices** Price breakdown, a JSON-serialized list of components (e.g. product price, tax, discount, delivery cost, delivery tax, bonus, etc.)
- 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
- **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
- **need\_phone\_number** Pass True if you require the user's phone number to complete the order
- need\_email Pass True if you require the user's email address to complete the order
- **need\_shipping\_address** Pass True if you require the user's shipping address to complete the order
- **send\_phone\_number\_to\_provider** Pass True if the user's phone number should be sent to provider
- **send\_email\_to\_provider** Pass True if the user's email address should be sent to provider
- is\_flexible Pass True if the final price depends on the shipping method
- 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** 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

answer\_invoice(title: str, description: str, payload: str, provider\_token: str, currency: str, prices:

List[LabeledPrice], max\_tip\_amount: Optional[int] = None, suggested\_tip\_amounts:

Optional[List[int]] = None, 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')>, 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
- 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 for your internal processes.
- **provider\_token** Payment provider token, obtained via @BotFather
- currency Three-letter ISO 4217 currency code, see more on currencies
- **prices** Price breakdown, a JSON-serialized list of components (e.g. product price, tax, discount, delivery cost, delivery tax, bonus, etc.)
- 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

- **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
- need\_phone\_number Pass True if you require the user's phone number to complete the order
- need\_email Pass True if you require the user's email address to complete the order
- **need\_shipping\_address** Pass True if you require the user's shipping address to complete the order
- **send\_phone\_number\_to\_provider** Pass True if the user's phone number should be sent to provider
- **send\_email\_to\_provider** Pass True if the user's email address should be sent to provider
- is\_flexible Pass True if the final price depends on the shipping method
- **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** 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

```
 \begin{aligned} \textbf{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]] = &Oefault('protect\_content') >, reply\_parameters: \\ Optional[ReplyParameters] = None, reply\_markup: \\ Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None, allow\_sending\_without\_reply: Optional[bool] = None, **kwargs: Any) &OendLocation  \end{aligned}
```

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

- chat\_id
- message\_thread\_id
- business\_connection\_id
- reply\_to\_message\_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
- 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
- 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

# 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')>, 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 <code>aiogram.methods.send\_location</code>. <code>SendLocation</code> 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

### **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
- 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

 $\begin{aligned} \textbf{reply\_media\_group}(\textit{media: List[Union[InputMediaAudio, InputMediaDocument, InputMediaPhoto,} \\ & \textit{InputMediaVideo]}, \textit{disable\_notification: Optional[bool]} = \textit{None, protect\_content:} \\ & \textit{Optional[Union[bool, Default]]} = & \textit{Optional[vpotect\_content')}, \textit{reply\_parameters:} \\ & \textit{Optional[ReplyParameters]} = \textit{None, allow\_sending\_without\_reply: Optional[bool]} = \\ & \textit{None, **kwargs: Any)} \rightarrow \textit{SendMediaGroup} \end{aligned}$ 

Shortcut for method  $aiogram.methods.send\_media\_group.SendMediaGroup$  will automatically fill method attributes:

- chat id
- message\_thread\_id
- business\_connection\_id
- reply\_to\_message\_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
- 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

## Returns

instance of method aiogram.methods.send\_media\_group.SendMediaGroup

Shortcut for method <code>aiogram.methods.send\_media\_group.SendMediaGroup</code> 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
- 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

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')>, 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')>, \*\*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
- reply\_to\_message\_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
- 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

### Returns

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')>, 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
- 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*
- $\bullet \ \ \textbf{link\_preview\_options} Link \ preview \ generation \ options \ for \ the \ message \\$
- **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
- **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

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, has\_spoiler: Optional[bool] = None, 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, \*\*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\_to\_message\_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*
- 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
- 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

### Returns

instance of method aiogram.methods.send\_photo.SendPhoto

answer\_photo(photo: Union[InputFile, str], caption: Optional[str] = None, parse\_mode:

Optional[Union[str, Default]] = <Default('parse\_mode')>, caption\_entities:

Optional[List[MessageEntity]] = None, has\_spoiler: Optional[bool] = None,
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) → SendPhoto

Shortcut for method aiogram.methods.send\_photo.SendPhoto 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*
- 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
- **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\_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')>, reply\_parameters: Optional[ReplyParameters] = 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\_to\_message\_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
- **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
- 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

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')>, 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

# **Parameters**

- 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
- 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

 $\begin{tabular}{l} \textbf{reply\_dice}(emoji: Optional[str] = None, disable\_notification: Optional[bool] = None, protect\_content: \\ Optional[Union[bool, Default]] = & Optional('protect\_content') >, reply\_parameters: \\ Optional[ReplyParameters] = None, reply\_markup: Optional[Union[InlineKeyboardMarkup, ReplyKeyboardMarkup, ReplyKeyboardRemove, ForceReply]] = None, \\ allow\_sending\_without\_reply: Optional[bool] = None, **kwargs: Any) \rightarrow SendDice \\ \end{tabular}$ 

 $Shortcut\ for\ method\ aiogram.methods.send\_dice.\ SendDice\ will\ automatically\ fill\ method\ attributes:$ 

- chat\_id
- message\_thread\_id
- business\_connection\_id
- reply\_to\_message\_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
- 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

### Returns

instance of method aiogram.methods.send\_dice.SendDice

answer\_dice(emoji: Optional[str] = None, 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) → SendDice

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
- **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 aiogram.methods.send\_sticker.SendSticker will automatically fill method attributes:

- chat id
- message\_thread\_id
- business\_connection\_id
- reply\_to\_message\_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.
- $\bullet$  **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
- **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

## Returns

instance of method aiogram.methods.send\_sticker.SendSticker

Shortcut for method aiogram.methods.send\_sticker.SendSticker 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
- 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 aiogram.methods.send\_venue.SendVenue will automatically fill method attributes:

- chat\_id
- message\_thread\_id
- business\_connection\_id
- reply\_to\_message\_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
- 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

### Returns

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')>, 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
- 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

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, has\_spoiler: Optional[bool] = None, supports\_streaming: Optional[bool] = None, 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, \*\*kwargs: Any) → SendVideo

Shortcut for method aiogram.methods.send\_video.SendVideo will automatically fill method attributes:

- chat id
- message\_thread\_id
- business\_connection\_id
- reply\_to\_message\_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* »
- 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*
- 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
- 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

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, has\_spoiler: Optional[bool] = None, supports\_streaming: Optional[bool] = None, 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) → SendVideo

Shortcut for method <code>aiogram.methods.send\_video.SendVideo</code> 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 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* »
- 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*
- 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
- 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\_note.SendVideoNote will automatically fill method attributes:

- · chat id
- message\_thread\_id
- business\_connection\_id
- reply\_to\_message\_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
- 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

### Returns

instance of method aiogram.methods.send\_video\_note.SendVideoNote

Shortcut for method  $aiogram.methods.send\_video\_note.SendVideoNote$  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
- 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')>, reply_parameters: Optional[ReplyParameters] = 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\_to\_message\_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* »
- 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
- 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

### 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')>, 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 aiogram.methods.send\_voice.SendVoice 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 <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* »
- 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
- 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 | None = None) → ForwardMessage | SendAnimation | 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

**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 -
- parse\_mode -

# 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, 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, 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*
- **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

# Returns

instance of method aiogram.methods.copy\_message.CopyMessage

edit\_text(text: str, inline\_message\_id: Optional[str] = None, parse\_mode: Optional[Union[str, Default]]
= <Default('parse\_mode')>, entities: Optional[List[MessageEntity]] = None,
link\_preview\_options: Optional[LinkPreviewOptions] = None, reply\_markup:
Optional[InlineKeyboardMarkup] = None, disable\_web\_page\_preview: Optional[Union[bool,
Default]] = <Default('link\_preview\_is\_disabled')>, \*\*kwargs: Any) → EditMessageText

Shortcut for method  $aiogram.methods.edit\_message\_text.EditMessageText$  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.Message</code> is returned, otherwise True is returned.

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

# **Parameters**

- text New text of the message, 1-4096 characters after entities parsing
- **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

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 <code>aiogram.methods.forward\_message.ForwardMessage</code> 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

 $instance\ of\ method\ aiogram. \verb|methods.forward_message|. Forward \verb|Message|$ 

edit\_media(media: InputMediaAnimation | InputMediaDocument | InputMediaAudio | InputMediaPhoto | InputMediaVideo, 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.

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

## **Parameters**

• media – A JSON-serialized object for a new media content of the message

- **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**(*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 aiogram.types.message.Message is returned, otherwise True is returned.

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

### **Parameters**

- **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(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 aiogram.types.message.Message is returned, otherwise True is returned.

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

### Parameters

**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
```

- 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 True is returned.

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

### **Parameters**

- latitude Latitude of new location
- longitude Longitude of new location
- 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**

instance of method aiogram.methods.edit\_message\_live\_location.
EditMessageLiveLocation

**stop\_live\_location**( $inline\_message\_id$ :  $str \mid None = None$ ,  $reply\_markup$ : InlineKeyboardMarkup | None = None, \*\*kwargs: Any)  $\rightarrow StopMessageLiveLocation$ 

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**

- **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.stop\_message\_live\_location.
StopMessageLiveLocation

edit\_caption(inline\_message\_id: Optional[str] = None, caption: Optional[str] = None, parse\_mode: Optional[Union[str, Default]] =  $\langle Default('parse\_mode') \rangle$ , caption\_entities: Optional[List[MessageEntity]] = None, reply\_markup: Optional[InlineKeyboardMarkup] = None, \*\*kwargs: Any)  $\rightarrow EditMessageCaption$ 

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 aiogram.types.message.Message is returned, otherwise True is returned.

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

### **Parameters**

- **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*
- reply\_markup A JSON-serialized object for an inline keyboard.

# Returns

 $instance\ of\ method\ aiogram.\ methods.edit\_message\_caption.\ EditMessageCaption$ 

 $delete(**kwargs: Any) \rightarrow DeleteMessage$ 

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

 $pin(disable\_notification: bool \mid None = None, **kwargs: Any) \rightarrow PinChatMessage$ 

Shortcut for method  $aiogram.methods.pin\_chat\_message.PinChatMessage$  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**

**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

**unpin**(\*\*kwargs: Any)  $\rightarrow UnpinChatMessage$ 

Shortcut for method <code>aiogram.methods.unpin\_chat\_message.UnpinChatMessage</code> 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

# Returns

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

```
get\_url(force\_private: bool = False) \rightarrow str \mid 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

### Returns

string with full message URL

react(reaction: List[ReactionTypeEmoji | ReactionTypeCustomEmoji] | 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. 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.
- is\_big Pass True to set the reaction with a big animation

### **Returns**

instance of method aiogram.methods.set\_message\_reaction.SetMessageReaction

# MessageAutoDeleteTimerChanged

class aiogram.types.message\_auto\_delete\_timer\_changed.MessageAutoDeleteTimerChanged(\*,

```
mes-
sage_auto_delete_time:
int,
**ex-
tra_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(\_ModelMetaclass\_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), '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.

 $model_post_init(ModelMetaclass context: Any) \rightarrow None$ 

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)  $\rightarrow str$ 

## 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.

 $model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None$ 

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

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

# 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(\_ModelMetaclass\_context: Any) \rightarrow None
```

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

# MessageOriginChannel

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(\_ModelMetaclass\_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

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(ModelMetaclass 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

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

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(\_ModelMetaclass\_context: Any) \rightarrow None
```

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

\*\*extra\_data: Any)

### 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, 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.

 $model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None$ 

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:
datetime,
reactions:
List[ReactionCount],
\*\*extra\_data:
Any)

This object represents reaction changes on a message with anonymous reactions.

 $Source: \ https://core.telegram.org/bots/api\#messagereaction countup dated$ 

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(\_ModelMetaclass\_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

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]

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.

 $model\_post\_init(\_\mathit{ModelMetaclass}\_\mathit{context} : \mathit{Any}) \rightarrow None$ 

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

new\_reaction: List[ReactionTypeEmoji | ReactionTypeCustomEmoji]

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

## **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(\_ModelMetaclass\_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 | 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
```

2.3. Bot API 211

total\_voter\_count: int

True, if the poll is closed

is\_closed: bool

Total number of users that voted in the poll

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.

 $model_post_init(ModelMetaclass context: Any) \rightarrow None$ 

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.

 $model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None$ 

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**

```
class aiogram.types.poll_option.PollOption(*, text: str, voter_count: int, text_entities: List[MessageEntity] | None = None, **extra_data: Any)

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

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

```
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(\_ModelMetaclass\_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(\_ModelMetaclass\_\_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**

type: ReactionTypeEmoji | ReactionTypeCustomEmoji
 Type of the reaction

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

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

 $model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None$ 

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

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(_ModelMetaclass__context: Any) → 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(\_ModelMetaclass\_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(\_ModelMetaclass\_context: Any) \rightarrow None
```

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

```
emoji: str
```

### 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(ModelMetaclass 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 <code>one\_time\_keyboard</code> in <code>aiogram.types.reply\_keyboard\_markup.ReplyKeyboardMarkup</code>)

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

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

```
model_post_init(\_ModelMetaclass\_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: ~typ-ing.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(ModelMetaclass 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.

```
{\bf model\_post\_init}(\_{\it ModelMetaclass\_context}. \ {\it Any}) \rightarrow {\it None}
```

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

```
retry_after: int | None
```

*Optional*. In case of exceeding flood control, the number of seconds left to wait before the request can be repeated

## **SharedUser**

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(\_ModelMetaclass\_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(\_ModelMetaclass\_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**

 $\textbf{class} \ \texttt{aiogram.types.switch\_inline\_query\_chosen\_chat.} \textbf{SwitchInlineQueryChosenChat} (*, \textit{query: str}) \\$ 

| *None* = None, allow\_user\_chats: bool | None = None, allow\_bot\_chats: bool | None = None, allow group chats: bool | None = None, allow 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(\_ModelMetaclass\_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**

**class** aiogram.types.text\_quote.**TextQuote**(\*, text: str, position: int, entities: List[MessageEntity] | None = None, is\_manual: bool | None = None, \*\*extra\_data: Any)

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(ModelMetaclass 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.

 $model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None$ 

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. 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 \mid None = None, limit: int \mid None = None, **kwargs: Any) \rightarrow GetUserProfilePhotos
```

Shortcut for method <code>aiogram.methods.get\_user\_profile\_photos.GetUserProfilePhotos</code> 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.

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

#### **Parameters**

- **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(\_ModelMetaclass\_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.

 $model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None$ 

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(\_ModelMetaclass\_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**

```
class aiogram.types.users_shared.UsersShared(*, request_id: int, users: List[SharedUser], user_ids:
List[int] | None = None, **extra_data: Any)
```

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(\_ModelMetaclass\_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 \mid 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(ModelMetaclass 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

```
class aiogram.types.video.Video(*, file_id: str, file_unique_id: str, width: int, height: int, duration: int, thumbnail: PhotoSize | None = None, file_name: str | None = None, mime_type: str | None = None, file_size: int | None = None, **extra_data: Any)
```

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 sender

## height: int

Video height as defined by sender

### duration: int

Duration of the video in seconds as defined by sender

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

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

```
model_post_init(_ModelMetaclass__context: Any) → 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 sender

## mime\_type: str | None

Optional. MIME type of the file as defined by 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(\_ModelMetaclass\_context: Any) \rightarrow None
```

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

# VideoChatParticipantsInvited

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(\_ModelMetaclass\_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(\_ModelMetaclass\_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(\_ModelMetaclass\_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 sender

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

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

```
model_post_init(_ModelMetaclass__context: Any) → 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 sender

thumbnail: PhotoSize | None
 Optional. Video thumbnail
file\_size: int | None

Optional. File size in bytes

### Voice

**class** aiogram.types.voice.**Voice**(\*, file\_id: str, file\_unique\_id: str, duration: int, mime\_type: str | None = None, file\_size: int | None = None, \*\*extra\_data: Any)

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 sender

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

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

```
model_post_init(\_ModelMetaclass\_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 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(ModelMetaclass 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(\_ModelMetaclass\_\_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(\_ModelMetaclass\_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(\_ModelMetaclass\_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(\_ModelMetaclass\_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

| InlineOueryResultArticle | InlineOueryResultAudio | InlineOueryResultContact |

InlineQueryResultGame | InlineQueryResultDocument | InlineQueryResultGif |

InlineOueryResultLocation | InlineOueryResultMpeg4Gif | InlineOueryResultPhoto |

InlineQueryResultVenue | InlineQueryResultVideo | InlineQueryResultVoice], cache\_time: int |

None = None, is\_personal: bool | None = None,  $next_offset$ : str | None = None, button:

InlineQueryResultsButton | None = None,  $switch\_pm\_parameter$ : str | None = None,  $switch\_pm\_text$ : str | None = None, \*\*kwargs: Any)  $\rightarrow AnswerInlineQuery$ 

Shortcut for method <code>aiogram.methods.answer\_inline\_query.AnswerInlineQuery</code> 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
- aiogram.types.inline\_query\_result\_cached\_mpeg4\_gif.InlineQueryResultCachedMpeg4Gif
- $\bullet \ aiogram.\ types.\ in line\_query\_result\_cached\_photo.\ In line\ Query\ Result\ Cached\ Photo$
- 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(\_ModelMetaclass\_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-

= InlineQueryResult-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

eral[InlineQueryResultType.ARTICLE]

| None = None, thumbnail\_height: int | 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(ModelMetaclass 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

```
class aiogram.types.inline_query_result_audio.InlineQueryResultAudio(*, type: ~typ-
                                                                                      ing.Literal[InlineQueryResultType.AUDIO]
                                                                                      = InlineQueryResult-
                                                                                      Type.AUDIO, id: str,
                                                                                      audio_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, 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.AUDI0]
     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.
```

```
model_post_init(_ModelMetaclass__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 | InputInvoiceMessageContent |
None
```

Optional. Content of the message to be sent instead of the audio

# InlineQueryResultCachedAudio

```
class aiogram.types.inline_query_result_cached_audio.InlineQueryResultCachedAudio(*, type:
```

```
~typ-
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(_ModelMetaclass__context: Any) → 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

```
class aiogram.types.inline_query_result_cached_document.InlineQueryResultCachedDocument(*,
                                                                                                           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
```

238 Chapter 2. Contents

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.

# $model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None$

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,
                                                                                                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 =
```

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

None,

\*\*extra\_data: ~typing.Any)

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.

 $model_post_init(\_ModelMetaclass\_\_context: Any) \rightarrow None$ 

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

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

```
class aiogram.types.inline_query_result_cached_mpeg4_gif.InlineQueryResultCachedMpeg4Gif(*,
                                                                                                           type:
                                                                                                           ing.Literal[InlineQ
                                                                                                           In-
                                                                                                           line-
                                                                                                           QueryRe-
                                                                                                           sult-
                                                                                                           Type.MPEG4_GIF
                                                                                                           id:
                                                                                                           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,
                                                                                                           re-
                                                                                                           ply_markup:
                                                                                                           ~aiogram.types.ini
                                                                                                           None
                                                                                                           None,
                                                                                                           put_message_cont
                                                                                                           ~aiogram.types.inp
```

~aiogram.types.inp

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(\_ModelMetaclass\_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

```
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
= Inline-
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, re-
ply_markup:
~aiogram.types.inline_keyb
| None =
None, in-
put_message_content:
~aiogram.types.input_text_i
~aiogram.types.input_locate
~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 photo stored on the Telegram servers. By default, this photo 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 photo.

```
Source: https://core.telegram.org/bots/api#inlinequeryresultcachedphoto
type: Literal[InlineQueryResultType.PHOTO]
     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
model_computed_fields: ClassVar[dict[str, ComputedFieldInfo]] = {}
     A dictionary of computed field names and their corresponding ComputedFieldInfo objects.
model_post_init(ModelMetaclass 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 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
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

246

```
{\bf class} \ {\bf aiogram.types.inline\_query\_result\_cached\_sticker.} {\bf InlineQueryResultCachedSticker} (*, {\bf class}) {\bf class} \ {\bf class
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 type:
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 Lit-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 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-

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.

```
\textbf{model\_post\_init}(\_\textit{ModelMetaclass\_\_context}: \textit{Any}) \rightarrow \textit{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:
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 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.VIDEO]
     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
model_computed_fields: ClassVar[dict[str, ComputedFieldInfo]] = {}
     A dictionary of computed field names and their corresponding ComputedFieldInfo objects.
model_post_init(\_ModelMetaclass\_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 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
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:} \\
```

```
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_r
~aiogram.types.input_locate
~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. $model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None$ 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 |

Optional. Content of the message to be sent instead of the voice message

None

## 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.

 $model_post_init(\_ModelMetaclass\_\_context: Any) \rightarrow None$ 

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 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

class aiogram.types.inline\_query\_result\_document.InlineQueryResultDocument(\*, type: ~typ-

```
ing.Literal[InlineQueryResultType.D
= 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

 $None = None, **extra\_data:$ 

Any)

```
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(\_ModelMetaclass\_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 |
```

## 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.

```
{\bf model\_post\_init}(\_{\it ModelMetaclass\_context}: {\it Any}) \rightarrow {\it 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

```
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
                                                                             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, 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 input_message_content 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
```

class aiogram.types.inline\_query\_result\_gif.InlineQueryResultGif(\*, type: ~typ-

2.3. Bot API 257

id: str

aif url: str

thumbnail\_url: str

gif\_width: int | None

gif\_height: int | None

Optional. Width of the GIF

Optional. Height of the GIF

Unique identifier for this result, 1-64 bytes

A valid URL for the GIF file. File size must not exceed 1MB

URL of the static (JPEG or GIF) or animated (MPEG4) thumbnail for the result

```
gif_duration: int | None
     Optional. Duration of the GIF in seconds
model_computed_fields: ClassVar[dict[str, ComputedFieldInfo]] = {}
     A dictionary of computed field names and their corresponding ComputedFieldInfo objects.
model_post_init(ModelMetaclass context: Any) \rightarrow None
     We need to both initialize private attributes and call the user-defined model_post_init method.
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'
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
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

## **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:  $int \mid None = None,$ \*\*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.

 $model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None$ 

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

~typing.Any)

```
class aiogram.types.inline_query_result_mpeg4_gif.InlineQueryResultMpeg4Gif(*, type: ~typ-
                                                                                              ing.Literal[InlineQueryResultType.In]
                                                                                              = Inline-
                                                                                              QueryResult-
                                                                                              Type.MPEG4_GIF,
                                                                                              id: str,
                                                                                              mpeg4_url: str,
                                                                                              thumbnail_url:
                                                                                              str,
                                                                                              mpeg4_width:
                                                                                              int | 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,
                                                                                              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:
```

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
model_computed_fields: ClassVar[dict[str, ComputedFieldInfo]] = {}
     A dictionary of computed field names and their corresponding ComputedFieldInfo objects.
model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None
     We need to both initialize private attributes and call the user-defined model_post_init method.
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'
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
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,
                                                                                     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.PHOT0]
```

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(_ModelMetaclass__context: Any) → 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
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] = Inline Query Result-Type.VENUE, id: str, *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,reply\_markup: InlineKeyboardMarkup | None = None,input\_message\_content: InputTextMessageContent | InputLocationMessage-Content | Input Venue Message Content InputContactMessageCon-InputInvoiceMessageContent | 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
    Optional. 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(\_ModelMetaclass\_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**

~aiogram.types.input\_contact\_message\_cont

~aiogram.types.input\_invoice\_message\_cont

| None = None, \*\*extra\_data: ~typing.Any)

```
class aiogram.types.inline_query_result_video.InlineQueryResultVideo(*, type: ~type-
                                                                                     ing.Literal[InlineQueryResultType.VIDEO]
                                                                                     = InlineQueryResult-
                                                                                     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,
                                                                                     video_width: int | 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
```

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.VIDEO]

Type of the result, must be video

## id: str

Unique identifier for this result, 1-64 bytes

### video\_url: str

A valid URL for the embedded video player or video file

```
mime_type: str
     MIME type of the content of the video URL, 'text/html' or 'video/mp4'
thumbnail url: str
     URL of the thumbnail (JPEG only) for the video
title: str
     Title for the result
caption: str | 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(\_ModelMetaclass\_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
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
```

*Optional*. Content of the message to be sent instead of the video. This field is **required** if InlineQueryResultVideo is used to send an HTML-page as a result (e.g., a YouTube video).

## **InlineQueryResultVoice**

ing.Literal[InlineQueryResultType.VOICE]

```
= InlineQueryResult-
                                                                                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_conte
                                                                                ~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
will be sent by the user. Alternatively, you can use input_message_content to send a message with the specified
content instead of the the voice message.
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
```

class aiogram.types.inline\_query\_result\_voice.InlineQueryResultVoice(\*, type: ~typ-

2.3. Bot API 269

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

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

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

 $model_post_init(ModelMetaclass context: Any) \rightarrow None$ 

```
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**

 $\textbf{class} \ \, \textbf{aiogram.types.inline\_query\_results\_button.} \\ \textbf{InlineQueryResultsButton} (*, \textit{text: str}, \textit{web\_app: } \\ \textbf{app: } \\ \textbf{app:$ 

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(\_ModelMetaclass\_\_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.

 $model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None$ 

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

```
aiogram Documentation, Release 3.7.0
class aiogram.types.input_invoice_message_content.InputInvoiceMessageContent(*, title: str,
                                                                                              description: str,
                                                                                              payload: str,
                                                                                              provider_token:
                                                                                              str, currency:
                                                                                              str, prices:
                                                                                              List[LabeledPrice],
                                                                                              max_tip_amount:
                                                                                              int | None =
                                                                                              None, sug-
                                                                                              gested_tip_amounts:
                                                                                              List[int] | None
                                                                                              = None,
                                                                                              provider_data:
                                                                                              str \mid None =
                                                                                              None,
                                                                                              photo_url: str |
                                                                                              None = None,
                                                                                              photo_size: int |
                                                                                              None = None,
                                                                                              photo_width:
                                                                                              int | None =
                                                                                              None,
                                                                                              photo_height:
                                                                                              int | None =
                                                                                              None.
                                                                                              need_name:
                                                                                              bool \mid None =
                                                                                              None,
                                                                                              need_phone_number:
                                                                                              bool | None =
                                                                                              None,
                                                                                              need_email:
                                                                                              bool \mid None =
                                                                                              None,
                                                                                              need_shipping_address:
                                                                                              bool | None =
                                                                                              None,
                                                                                              send_phone_number_to_provider:
                                                                                              bool \mid 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

Any)

is\_flexible: bool
| None = None,
\*\*extra\_data:

### description: str

Product description, 1-255 characters

## payload: str

Bot-defined invoice payload, 1-128 bytes. This will not be displayed to the user, use for your internal processes.

## provider\_token: str

Payment provider token, obtained via @BotFather

#### currency: str

Three-letter ISO 4217 currency code, see more on currencies

## prices: List[LabeledPrice]

Price breakdown, a JSON-serialized list of components (e.g. product price, tax, discount, delivery cost, delivery tax, bonus, etc.)

## max\_tip\_amount: int | 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

## 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.

## $model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None$

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

### need\_phone\_number: bool | None

Optional. Pass True if you require the user's phone number to complete the order

```
need_email: bool | None
     Optional. Pass True if you require the user's email address to complete the order
need_shipping_address: bool | None
     Optional. Pass True if you require the user's shipping address to complete the order
send_phone_number_to_provider: bool | None
     Optional. Pass True if the user's phone number should be sent to provider
send_email_to_provider: bool | None
     Optional. Pass True if the user's email address should be sent to provider
is_flexible: bool | None
     Optional. Pass True if the final price depends on the shipping method
```

## 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.

 $model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None$ 

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
- 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(\_ModelMetaclass\_context: Any) \rightarrow None
```

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

## InputTextMessageContent

|None = < De-

\*\*extra\_data: ~typing.Any)

fault('disable web page preview')>,

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(\_ModelMetaclass\_context: Any) \rightarrow None
```

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

```
link_preview_options: LinkPreviewOptions | None
```

Optional. Link preview generation options for the message

```
disable_web_page_preview: bool | Default | 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(\_ModelMetaclass\_\_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(\_ModelMetaclass\_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(\_ModelMetaclass\_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(\_ModelMetaclass\_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

```
class aiogram.types.sticker.Sticker(*, file_id: str, file_unique_id: str, type: str, width: int, height: int,
                                            is_animated: bool, is_video: bool, thumbnail: PhotoSize | None =
                                            None, emoji: str \mid None = None, set\_name: str \mid None = None,
                                            premium_animation: File | None = None, mask_position:
                                             MaskPosition | None = None, custom_emoji_id: str | None = None,
                                            needs repainting: bool | None = None, file size: int | None = None,
                                             **extra data: Any)
     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(\_ModelMetaclass\_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
```

2.3. Bot API 279

Optional. For premium regular stickers, premium animation for the sticker

Optional. For mask stickers, the position where the mask should be placed

mask\_position: MaskPosition | None

#### 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

## $\mathtt{set\_position\_in\_set}(position: int, **kwargs: Any) \rightarrow SetStickerPositionInSet$

Shortcut for method aiogram.methods.set\_sticker\_position\_in\_set. 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

#### Returns

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**

is\_animated: bool | None = None, is\_video: bool | None = None, \*\*extra\_data: Any)

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.

### $model_post_init(ModelMetaclass context: Any) \rightarrow None$

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

## **Telegram Passport**

## **EncryptedCredentials**

Describes data required for decrypting and authenticating <code>aiogram.types.encrypted\_passport\_element.</code> 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(\_ModelMetaclass\_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**

 $\textbf{class} \ \texttt{aiogram.types.encrypted\_passport\_element.} \textbf{EncryptedPassportElement} (*, \textit{type: str}, \textit{hash: str}, \textit{type: str}, \textit{hash: str}, \textit{type: str}, \textit{hash: str}, \textit{type: str}, \textit{hash: str}, \textit{type: st$ 

 $data: str \mid None =$ *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

 $Base 64-encoded\ element\ hash\ for\ using\ in\ a iogram.\ types.passport\_element\_error\_unspecified.$  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(\_ModelMetaclass\_\_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 <code>aiogram.types.encrypted\_credentials.EncryptedCredentials</code>.

#### 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(\_ModelMetaclass\_\_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(\_\mathit{ModelMetaclass}\_\mathit{context} : \mathit{Any}) \rightarrow None
```

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

# **PassportElementErrorDataField**

 ${\bf class}\ {\bf aiogram.types.passport\_element\_error\_data\_field. \textbf{PassportElementErrorDataField}(*, and {\bf class}) aiogram.types.passport\_element\_error\_data\_field. \textbf{PassportElementErrorDataField}(*, {\bf class}) aiogram.ty$ 

```
source:
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(\_ModelMetaclass\_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]
=
PassportElementErrorType.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(\_ModelMetaclass\_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**

= PassportElementErrorType.FILES, type: str, file\_hashes: List[str], message: str, \*\*extra\_data: Any)

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.

## $model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None$

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**

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(ModelMetaclass 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(*,
```

```
source:
Lit-
eral[PassportEleme
Pass-
portEle-
mentEr-
rorType.REVERSE_
type:
str,
file_hash:
str,
mes-
sage:
str,
**ex-
tra data:
Any)
```

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(\_ModelMetaclass\_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(\_ModelMetaclass\_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(ModelMetaclass 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(\*,

eral[Po = PassportEld mentEld rorTyp type: str, file\_had List[std message:

> str, \*\*extra\_da Any)

> source Lit-

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(\_ModelMetaclass\_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(\_ModelMetaclass\_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(\_ModelMetaclass\_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

## **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(\_ModelMetaclass\_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

#### 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(\_ModelMetaclass\_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.OrderInfo(*, name: str \mid None = None, phone_number: str \mid None = None, email: str \mid None = None, shipping_address: ShippingAddress \mid 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(\_ModelMetaclass\_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

## **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

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(\_ModelMetaclass\_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

## **ShippingAddress**

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

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

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

 $model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None$ 

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**

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(\_ModelMetaclass\_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(ModelMetaclass 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) <math>\rightarrow 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

```
instance of method aiogram.methods.answer_shipping_query.
AnswerShippingQuery
```

## 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

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.

 $model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None$ 

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

## **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, 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 non-service 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.

## $model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None$

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

## 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

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.

 $model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None$ 

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(\_ModelMetaclass\_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(_ModelMetaclass__context: Any) → 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 aiogram.methods.set\_game\_score.SetGameScore, or manually edited using aiogram.methods.edit\_message\_text.EditMessageText. 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**

This object represents one row of the high scores table for a game. And that's about all we've got for now.

If you've got any questions, please check out our https://core.telegram.org/bots/faq Bot FAQ »

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

## position: int

Position in high score table for the game

user: *User*User

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

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

```
model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None
```

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

score: int
Score

# 2.3.4 Methods

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(\_ModelMetaclass\_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(\_ModelMetaclass\_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(...)
```

#### Imports:

- from aiogram.methods.create\_new\_sticker\_set import CreateNewStickerSet
- alias: from aiogram.methods import CreateNewStickerSet

## With specific bot

```
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(\_ModelMetaclass\_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(...)
```

#### Imports:

- from aiogram.methods.delete\_sticker\_from\_set import DeleteStickerFromSet
- alias: from aiogram.methods import DeleteStickerFromSet

## With specific bot

```
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(\_ModelMetaclass\_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(...)
```

#### Imports:

- from aiogram.methods.delete\_sticker\_set import DeleteStickerSet
- alias: from aiogram.methods import DeleteStickerSet

## With specific bot

```
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(\_ModelMetaclass\_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(...)
```

#### Imports:

- from aiogram.methods.get\_custom\_emoji\_stickers import GetCustomEmojiStickers
- alias: from aiogram.methods import GetCustomEmojiStickers

## With specific bot

```
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(\_ModelMetaclass\_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:

308

- 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**(\_*ModelMetaclass\_\_context: Any*) → 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(...)
```

#### Imports:

- from aiogram.methods.replace\_sticker\_in\_set import ReplaceStickerInSet
- alias: from aiogram.methods import ReplaceStickerInSet

## With specific bot

```
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 | 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')>, 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.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(\_ModelMetaclass\_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

## 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_sticker(...)
```

#### Method as object

#### Imports:

- from aiogram.methods.send\_sticker import SendSticker
- alias: from aiogram.methods import SendSticker

# With specific bot

```
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()

# setCustomEmojiStickerSetThumbnail

Returns: bool

class aiogram.methods.set\_custom\_emoji\_sticker\_set\_thumbnail.SetCustomEmojiStickerSetThumbnail(\*,

str,
custom\_emoji
str
|
None
=
None,
\*\*extra\_data:
Any)

name:

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(ModelMetaclass 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.

## 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(\_ModelMetaclass\_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(\_ModelMetaclass\_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(\_ModelMetaclass\_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(\_ModelMetaclass\_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(\_ModelMetaclass\_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 32 kilobytes in size (see https://core.telegram.org/stickers#animatedup to sticker-requirements <a href="https://core.telegram.org/stickers#animated-sticker-">https://core.telegram.org/stickers#animated-sticker-</a> requirements>`\_`https://core.telegram.org/stickers#animated-sticker-requirements for animated sticker technical requirements), or a WEBM video with the thumbnail up to 32 kilobytes in size; see https://core.telegram.org/stickers#video-sticker-requirements <a href="https://core.telegram.org/stickers#video-sticker-requirements">https://core.telegram.org/stickers#video-sticker-requirements</a> sticker-requirements>`\_`https://core.telegram.org/stickers#video-sticker-requirements for video sticker technical requirements. Pass a file\_id as a String to send a file that already exists on the Telegram servers, pass an HTTP URL as a String for Telegram to get a file from the Internet, or upload a new one using multipart/form-data. More information on Sending Files ». Animated and video sticker set thumbnails can't be uploaded via HTTP URL. If omitted, then the thumbnail is dropped and the first sticker is used as the thumbnail.

## **Usage**

#### As bot method

```
result: bool = await bot.set_sticker_set_thumbnail(...)
```

# Method as object

Imports:

- 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(\_ModelMetaclass\_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(\_ModelMetaclass\_\_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(\_ModelMetaclass\_\_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(\_ModelMetaclass\_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(\_\mathit{ModelMetaclass}\_\mathit{context} : \mathit{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(\_ModelMetaclass\_\_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(\_ModelMetaclass\_\_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(\_ModelMetaclass\_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

```
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(\_ModelMetaclass\_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:

- $\bullet \ \, from \ \, aiogram.methods.close\_general\_forum\_topic \ \, import \ \, CloseGeneralForumTopic$
- alias: from aiogram.methods import CloseGeneralForumTopic

```
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
                                                          ~aiogram.client.default.Default | None =
                                                          <Default('parse_mode')>, caption_entities: ~typ-
                                                          ing.List[~aiogram.types.message_entity.MessageEntity]
                                                          | None = None, disable_notification: bool | 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 \mid None = None,
                                                          **extra_data: ~typing.Any)
```

Use this method to copy messages of any kind. Service 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(\_ModelMetaclass\_\_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

```
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: 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, 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</code>. ForwardMessages, but the copied messages don't have a link to the original message. Album grouping is kept for copied messages. On success, an array of <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(\_ModelMetaclass\_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(ModelMetaclass 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

#### **Usage**

#### 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()

### 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(\_ModelMetaclass\_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 aiogram.methods. get\_forum\_topic\_icon\_stickers.GetForumTopicIconStickers 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(\_ModelMetaclass\_\_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(\_ModelMetaclass\_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.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(\_ModelMetaclass\_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.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(\_ModelMetaclass\_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(\_ModelMetaclass\_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.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
```

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(\_ModelMetaclass\_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

```
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()

### 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 *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(\_ModelMetaclass\_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 *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(\_ModelMetaclass\_\_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(\_ModelMetaclass\_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(\_ModelMetaclass\_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(ModelMetaclass 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.

 $model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None$ 

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(\_ModelMetaclass\_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(\_ModelMetaclass\_context: Any) \rightarrow None
```

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

#### Usage

#### As bot method

# Method as object

#### Imports:

- from aiogram.methods.get\_chat\_administrators import GetChatAdministrators
- alias: from aiogram.methods import GetChatAdministrators

#### With specific bot

### 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(\_ModelMetaclass\_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: Union[ChatMemberOwner, ChatMemberAdministrator, ChatMemberMember, __ 

→ ChatMemberRestricted, ChatMemberLeft, ChatMemberBanned] = await bot.get_chat_member(...

→)
```

### 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(\_ModelMetaclass\_\_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(\_ModelMetaclass\_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 

→bot(GetChatMenuButton(...))
```

#### 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. <code>Note:</code> 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(\_ModelMetaclass\_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(\_ModelMetaclass\_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(\_ModelMetaclass\_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:

356

- from aiogram.methods.get\_me import GetMe
- alias: from aiogram.methods import GetMe

## With specific bot

```
result: User = await bot(GetMe(...))
```

## getMyCommands

returned.

```
Returns: List[BotCommand]
```

```
class aiogram.methods.get_my_commands.GetMyCommands(*, scope: BotCommandScopeDefault |
BotCommandScopeAllPrivateChats |
BotCommandScopeAllGroupChats |
```

BotCommandScopeAllChatAdministrators |

BotCommandScopeChat |

BotCommandScopeChatAdministrators |

 $BotCommandScopeChatMember \mid None = None,$ 

language\_code: str | 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

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(\_ModelMetaclass\_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(\_ModelMetaclass\_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(ModelMetaclass 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.

 $model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None$ 

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:

360

- 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.

 $model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None$ 

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(\_ModelMetaclass\_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(\_ModelMetaclass\_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(\_ModelMetaclass\_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(_ModelMetaclass__context: Any) → 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(\_ModelMetaclass\_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(\_ModelMetaclass\_context: Any) \rightarrow None
```

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

```
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.

#### 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 | None = None, can\_manage\_chat: bool | None = None, can\_delete\_messages: bool | None = None, can\_delete\_messages: bool | None = None, can\_manage\_video\_chats: bool | 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\_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.

## $model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None$

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(\_ModelMetaclass\_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(ModelMetaclass 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(\_ModelMetaclass\_\_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\_other\_messages*, *can\_send\_other\_messages*, and *can\_send\_other\_other\_messages*, and *can\_send\_other\_ot* 

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(\_ModelMetaclass\_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:

- $\bullet \ \, from \ \, aiogram.methods.revoke\_chat\_invite\_link \ \, import \ \, RevokeChatInviteLink \\$
- 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, has\_spoiler: bool | None = None,
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 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

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

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

```
model_post_init(\_ModelMetaclass\_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 animation 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

```
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

```
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_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()
- $\bullet \ aiogram.types.chat\_join\_request.ChatJoinRequest.answer\_animation\_pm()\\$
- aiogram.types.chat\_member\_updated.ChatMemberUpdated.answer\_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 = <Default('parse mode')>, 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')>, 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 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(ModelMetaclass 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

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_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()

### 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(\_ModelMetaclass\_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

~aiogram.types.reply\_keyboard\_remove.ReplyKeyboardRemove | ~aiogram.types.force\_reply.ForceReply | None = None, allow\_sending\_without\_reply: bool | 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(\_ModelMetaclass\_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

```
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:

- $\bullet \ \, \text{from alogram.methods.send\_contact import SendContact}$
- 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()

#### 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')>, 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.

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

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

```
model_post_init(\_ModelMetaclass\_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

```
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_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()

#### sendDocument

Returns: Message

class aiogram.methods.send\_document.SendDocument(\*, chat\_id: int | str, document:

```
~aiogram.types.input_file.InputFile | str,
business_connection_id: str | None = None,
message_thread_id: int | None = None, thumbnail:
\simaiogram.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, disable_content_type_detection:
bool | None = None, disable_notification: bool |
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 \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(\_ModelMetaclass\_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

```
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()

## 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')>, 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.

### $model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None$

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

```
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()

## sendMediaGroup

```
Returns: List[Message]
```

reply parameters:

~aiogram.types.reply\_parameters.ReplyParameters | 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 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(ModelMetaclass 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

```
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()

## 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')>, 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 \mid None = None,
```

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

\*\*extra\_data: ~typing.Any)

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: str

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

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

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

 $model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None$ 

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

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

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

#### 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()

### 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, has spoiler: bool | None = None,*disable notification: bool* | *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  $| \sim aiogram.types.force\_reply.ForceReply | None = None,$ 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 parse\_mode

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

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

```
model_post_init(\_ModelMetaclass\_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

```
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()

#### sendPoll

```
Returns: Message
```

```
class aiogram.methods.send_poll.SendPoll(*, chat_id: int | str, question: str, options: ~typ-
                                                    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 | <math>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')>, 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 a native poll. On success, the sent *aiogram.types.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

### 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(\_ModelMetaclass\_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

```
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()

#### 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 | None = None,
                                                     message thread id: int | None = None, foursquare id: str |
                                                     None = None, foursquare\_type: str | None = None,
                                                     google place id: str \mid None = None, google place type: str
                                                     | None = None, disable\_notification: bool | 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
                                                      | \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 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.

# $model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None$

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

### 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_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()

### 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 | None = None, parse_mode: str |
~aiogram.client.default.Default | None =
<Default('parse mode')>, caption entities:
~typing.List[~aiogram.types.message_entity.MessageEntity]
| None = None, has\_spoiler: bool | None = None,
supports\_streaming: bool \mid None = None,
disable_notification: bool | 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
| \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 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.

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

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

```
model_post_init(\_ModelMetaclass\_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

```
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

```
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_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()

### 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 | 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')>, 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.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(\_ModelMetaclass\_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

```
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:

- $\bullet \ \, from \ \, aiogram.methods.send\_video\_note \ \, import \ \, SendVideoNote$
- 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()

#### 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* = <*Default('parse\_mode')*>, *caption\_entities:* ~typing.List[~aiogram.types.message entity.MessageEntity] | None = None, duration: int | None = None, $disable\_notification: bool \mid None = None, protect\_content:$ 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  $| \sim aiogram.types.force\_reply.ForceReply | None = None,$  $allow\_sending\_without\_reply: bool | None = None,$ reply\_to\_message\_id: int | None = None, \*\*extra\_data: ~typing.Any)

Use this method to send 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>. 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

## 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

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

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

```
model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None
```

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

#### duration: int | None

Duration of the voice message in seconds

## 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: 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()

## setChatAdministratorCustomTitle

Returns: bool

class aiogram.methods.set\_chat\_administrator\_custom\_title.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(\_ModelMetaclass\_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(\_ModelMetaclass\_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

#### **Usage**

## 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

success.

Returns: bool

 $\textbf{class} \ \texttt{aiogram.methods.set\_chat\_menu\_button.SetChatMenuButton} (*, \textit{chat\_id: int} \mid \textit{None} = \textit{None}, \\$ 

*menu\_button:* 

MenuButtonCommands | MenuButtonWebApp |

 $MenuButtonDefault \mid None = None,$ 

\*\*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

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(ModelMetaclass 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 a $iogram.types.menu\_button\_default.MenuButtonDefault$ 

# **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(\_ModelMetaclass\_\_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.

#### 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(\_ModelMetaclass\_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

#### 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(\_ModelMetaclass\_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

#### 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(ModelMetaclass 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

#### 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

List[ReactionTypeEmoji |
ReactionTypeCustomEmoji] | 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. 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(\_ModelMetaclass\_context: Any) \rightarrow None
```

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

```
reaction: List[ReactionTypeEmoji | ReactionTypeCustomEmoji] | 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.

```
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 |
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(\_ModelMetaclass\_\_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

 $\textbf{class} \ \, \textbf{aiogram.methods.set\_my\_default\_administrator\_rights.} \\ \textbf{SetMyDefaultAdministratorRights}(*, \texttt{my\_default\_administrator\_rights}) \\ \textbf{aiogram.methods.set\_my\_default\_administrator\_rights}(*, \texttt{my\_default\_administrator\_rights}) \\ \textbf{aiogram.methods}(*, \texttt{my\_default\_administrator\_rights}) \\ \textbf{aiogram.me$ 

```
rights:
ChatAd-
min-
is-
tra-
tor-
Rights
None
None,
for_channels:
bool
None
None,
**ex-
tra_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(\_ModelMetaclass\_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.

### **Usage**

#### 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(\_\mathit{ModelMetaclass}\_\mathit{context} : \mathit{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.

### **Usage**

### 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(\_ModelMetaclass\_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.

## **Usage**

### 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(\_ModelMetaclass\_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(\_ModelMetaclass\_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(\_ModelMetaclass\_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()

# unhide General Forum Topic

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(\_ModelMetaclass\_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(\_ModelMetaclass\_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:

- $\bullet \ \ from \ aiogram.methods.unpin\_all\_chat\_messages \ import \ Unpin\verb|AllChatMessages|$
- 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

```
class aiogram.methods.unpin_all_forum_topic_messages.UnpinAllForumTopicMessages(*, chat\_id: int \mid str, mes-
sage\_thread\_id:
int, **ex-
tra\_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(\_ModelMetaclass\_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

 $\textbf{class} \ \ a iogram. \texttt{methods.unpin\_all\_general\_forum\_topic\_messages.} \textbf{UnpinAllGeneralForumTopicMessages} (*, \texttt{messages.unpinAllGeneralForumTopicMessages}) (*, \texttt{messages.unpinAllGeneralForumTopicMessag$ 

```
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(\_ModelMetaclass\_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

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)

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

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

```
model_post_init(\_ModelMetaclass\_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 a message to unpin. 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:

- $\bullet \ \, \text{from aiogram.methods.unpin\_chat\_message import UnpinChatMessage}$
- 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(\_ModelMetaclass\_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(\_ModelMetaclass\_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, 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, reply\_markup:

~aiogram.types.inline\_keyboard\_markup.InlineKeybo | None = None, \*\*extra\_data:

|None| = None, \*\*\*ex-

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.Message</code> is returned, otherwise <code>True</code> is returned.

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

```
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(\_ModelMetaclass\_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

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]

class aiogram.methods.edit\_message\_live\_location.EditMessageLiveLocation(\*, latitude: float,

longitude: float, 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 <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

latitude: float

Latitude of new location

longitude: float

Longitude of new location

#### 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(\_ModelMetaclass\_\_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]
```

class aiogram.methods.edit\_message\_media.EditMessageMedia(\*, media: InputMediaAnimation |

InputMediaDocument |
InputMediaAudio | InputMediaPhoto |
InputMediaVideo, 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 True is returned.

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

```
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(\_ModelMetaclass\_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.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(*, 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 <code>True</code> is returned.

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

```
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.

```
{\bf model\_post\_init}(\_{\it ModelMetaclass\_context}: {\it Any}) \rightarrow {\it 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, chat id: int | str | None = None,
                                                                    message id: int \mid None = None,
                                                                    inline message id: str \mid 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
                                                                    | None = None, 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 aiogram.types.message.Message is returned, otherwise True is returned.
     Source: https://core.telegram.org/bots/api#editmessagetext
     text: str
           New text of the message, 1-4096 characters after entities parsing
     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
     parse_mode: str | Default | None
           Mode for parsing entities in the message text. 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(\_ModelMetaclass\_context: Any) \rightarrow None
           We need to both initialize private attributes and call the user-defined model_post_init method.
     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 | 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]
```

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

```
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(\_ModelMetaclass\_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: Pol1
```

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(\_ModelMetaclass\_context: Any) \rightarrow None
```

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

```
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 |
InlineOueryResultCachedPhoto
InlineQueryResultCachedSticker
InlineQueryResultCachedVideo |
InlineQueryResultCachedVoice |
InlineQueryResultArticle |
InlineQueryResultAudio |
InlineOueryResultContact |
InlineQueryResultGame |
InlineQueryResultDocument |
InlineQueryResultGif |
InlineQueryResultLocation |
InlineQueryResultMpeg4Gif
InlineQueryResultPhoto |
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(\_ModelMetaclass\_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 <code>switch\_pm\_parameter</code>

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 | In line Query Result Cached Document| InlineOueryResultCachedGif | InlineQueryResultCachedMpeg4Gif | InlineQueryResultCachedPhoto | InlineQueryResultCachedSticker | InlineQueryResultCachedVideo | InlineQueryResultCachedVoice | InlineQueryResultArticle | InlineQueryResultAudio | InlineQueryResultContact | InlineQueryResultGame | InlineQueryResultDocument | InlineQueryResultGif | InlineQueryResultLocation | InlineQueryResultMpeg4Gif | InlineQueryResultPhoto | InlineQueryResultVenue | InlineQueryResultVideo | InlineOueryResultVoice. \*\*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</code>. <code>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(\_ModelMetaclass\_\_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(\_ModelMetaclass\_\_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')>, 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:
          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.
     model_computed_fields: ClassVar[dict[str, ComputedFieldInfo]] = {}
          A dictionary of computed field names and their corresponding ComputedFieldInfo objects.
     model_post_init(ModelMetaclass 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
     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
          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
```

2.3. Bot API 453

Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023

If the message is a reply, ID of the original message

## **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()

#### 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 *aiogram.types.message.Message* 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 *force* is False.

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(\_ModelMetaclass\_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 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

# 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(\_ModelMetaclass\_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(\_ModelMetaclass\_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, provider\_token: str, currency: str, prices: List[LabeledPrice],  $max\_tip\_amount: int \mid None = None,$ suggested tip amounts: List[int] | None = None, provider data: strNone = None, photo url: str | None =*None*,  $photo\_size$ :  $int \mid None = None$ , photo width: int | None = None,  $photo\_height: int \mid None = None,$  $need\_name: bool \mid None = None,$ 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 for your internal processes.

### provider\_token: str

Payment provider token, obtained via BotFather

#### currency: str

Three-letter ISO 4217 currency code, see more on currencies

# prices: List[LabeledPrice]

Price breakdown, a JSON-serialized list of components (e.g. product price, tax, discount, delivery cost, delivery tax, bonus, etc.)

#### max\_tip\_amount: int | 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

# 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(\_ModelMetaclass\_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

need\_phone\_number: bool | None

Pass True if you require the user's phone number to complete the order

need\_email: bool | None

Pass True if you require the user's email address to complete the order

need\_shipping\_address: bool | None

Pass True if you require the user's shipping address to complete the order

send\_phone\_number\_to\_provider: bool | None

Pass True if the user's phone number should be sent to the provider

send\_email\_to\_provider: bool | None

Pass True if the user's email address should be sent to the provider

is\_flexible: bool | None

Pass True if the final price depends on the shipping method

# 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(...)
```

#### sendInvoice

Returns: Message

message\_thread\_id: int | None = None,
max\_tip\_amount: int | None = None,
suggested\_tip\_amounts: ~typing.List[int] | None =
None, start\_parameter: str | None = None,
provider\_data: str | None = None, photo\_url: str | None
= None, photo\_size: int | None = None, photo\_width:

ing.List[~aiogram.types.labeled\_price.LabeledPrice],

int | None = None, photo\_height: int | None = None,
need\_name: bool | None = None, need\_phone\_number:
bool | None = None, need\_email: bool | None = None,

need\_shipping\_address: bool | None = None,

send\_phone\_number\_to\_provider: bool | None = None, send\_email\_to\_provider: bool | None = None,

is\_flexible: bool | None = None, disable\_notification:

is\_flexible: bool | None = None, disable\_notification bool | 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

| 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 for your internal processes.

#### provider\_token: str

Payment provider token, obtained via @BotFather

#### currency: str

Three-letter ISO 4217 currency code, see more on currencies

#### prices: List[LabeledPrice]

Price breakdown, a JSON-serialized list of components (e.g. product price, tax, discount, delivery cost, delivery tax, bonus, etc.)

### message\_thread\_id: int | None

Unique identifier for the target message thread (topic) of the forum; for forum supergroups only

#### 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

# 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.

 $model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None$ 

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

need\_phone\_number: bool | None

Pass True if you require the user's phone number to complete the order

need\_email: bool | None

Pass True if you require the user's email address to complete the order

need\_shipping\_address: bool | None

Pass True if you require the user's shipping address to complete the order

send\_phone\_number\_to\_provider: bool | None

Pass True if the user's phone number should be sent to provider

send\_email\_to\_provider: bool | None

Pass True if the user's email address should be sent to provider

is\_flexible: bool | None

Pass True if the final price depends on the shipping 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

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()

#### **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(\_ModelMetaclass\_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(\_ModelMetaclass\_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(\_ModelMetaclass\_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}. \textbf{SetWebhook}(*, \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 = None, \  \, \textit{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: \  \, \textit{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.

## $model_post_init(\_ModelMetaclass\_context: Any) \rightarrow None$

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 <code>aiogram.types.update.Update</code> for a complete list of available update types. Specify an empty list to receive all update types except <code>chat\_member</code>, <code>message\_reaction</code>, and <code>message\_reaction\_count</code> (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,  $\emptyset-9$ , \_ 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(\_ModelMetaclass\_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**

**class** aiogram.enums.chat\_action.**ChatAction**(*value*, *names=None*, \*, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

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

```
\begin{tabular}{ll} \textbf{class} & aiogram.enums.chat\_member\_status.\textbf{ChatMemberStatus}(value, names=None, *, module=None, qualname=None, type=None, start=1, boundary=None) \end{tabular}
```

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'
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'
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'

BRL = 'BRL'
BYN = 'BYN'

BOB = 'BOB'

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**

```
class aiogram.enums.dice_emoji.DiceEmoji(value, names=None, *, module=None, qualname=None, type=None, start=1, boundary=None)
```

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#encryptedpassportelement

```
PERSONAL_DETAILS = 'personal_details'

PASSPORT = 'passport'

DRIVER_LICENSE = 'driver_license'

IDENTITY_CARD = 'identity_card'

INTERNAL_PASSPORT = 'internal_passport'

ADDRESS = 'address'

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'
```

2.3. Bot API 479

VENUE = 'venue'

### InputMediaType

## KeyboardButtonPollTypeType

```
class aiogram.enums.keyboard_button_poll_type_type.KeyboardButtonPollTypeType(value,
```

```
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

## MessageEntityType

WEB\_APP = 'web\_app'

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'
CODE = 'code'
PRE = 'pre'
TEXT_LINK = 'text_link'
TEXT_MENTION = 'text_mention'
CUSTOM_EMOJI = 'custom_emoji'
```

### MessageOriginType

#### **ParseMode**

Formatting options

CHANNEL = 'channel'

Source: https://core.telegram.org/bots/api#formatting-options

```
MARKDOWN_V2 = 'MarkdownV2'
MARKDOWN = 'Markdown'
HTML = 'HTML'
```

## **PassportElementErrorType**

This object represents a passport element error type.

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
```

```
class aiogram.enums.poll_type.PollType(value, names=None, *, module=None, qualname=None,
                                          type=None, start=1, boundary=None)
```

This object represents poll type

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

```
REGULAR = 'regular'
QUIZ = 'quiz'
```

## ReactionTypeType

```
class aiogram.enums.reaction_type_type.ReactionTypeType(value, names=None, *, module=None,
                                                             qualname=None, type=None, start=1,
                                                             boundary=None)
```

This object represents reaction type.

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

```
EMOJI = 'emoji'
CUSTOM_EMOJI = 'custom_emoji'
```

#### **StickerFormat**

```
class aiogram.enums.sticker_format.StickerFormat(value, names=None, *, module=None,
                                                     qualname=None, type=None, start=1,
                                                     boundary=None)
```

Format of the sticker

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

```
STATIC = 'static'
ANIMATED = 'animated'
VIDEO = 'video'
```

# **StickerType**

```
class aiogram.enums.sticker_type.StickerType(value, names=None, *, module=None, qualname=None,
                                                 type=None, start=1, boundary=None)
```

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
```

## **UpdateType**

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'

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'
```

#### 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 \mid Path \mid str \mid None = None, timeout: int = 30, chunk_size: int = 65536, seek: bool = True) <math>\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.BytesI0 = 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

## Warning: Be respectful with Telegram

Instances of *InputFile* are reusable. That's mean you can create instance of InputFile and sent this file multiple times but Telegram does not recommend to do that and when you upload file once just save their *file\_id* and use it in next times.

## 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"
)
```

# 2.4 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) → Router
    Attach another router.
    Parameters
```

Returns

router -

```
include_routers(*routers: Router) → None
   Attach multiple routers.

Parameters
    routers -

Returns

resolve_used_update_types(skip_events: Set[str] | None = None) → List[str]
   Resolve registered event names

Is useful for getting updates only for registered event types.

Parameters
    skip_events - skip specified event names

Returns
    set of registered names
```

#### **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

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**

#### 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

## Message reaction count

#### **Chat boost**

```
@router.chat_boost()
async def chat_boost_handler(chat_boost: types.ChatBoostUpdated) -> Any: pass
```

#### Remove chat boost

#### **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**

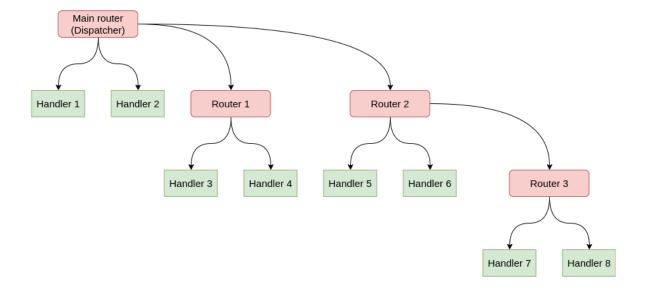
```
@dispatcher.update()
async def message_handler(update: types.Update) -> Any: pass
```

Warning: The only root Router (Dispatcher) can handle this type of event.

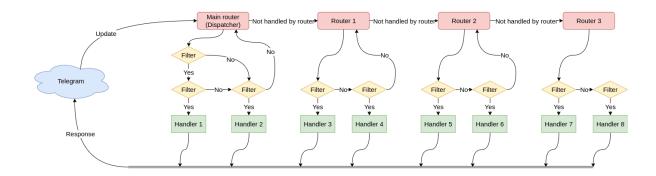
**Note:** Dispatcher already has default handler for this event type, so you can use it for handling all updates that are not handled by any other handlers.

### 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

**async feed\_raw\_update**(*bot: Bot, update: Dict[str, Any], \*\*kwargs: Any*) → 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 Dispatcher.feed\_update(bot=..., update=...) method:

```
bot = Bot(...)
dp = Dispatcher()
...
result = await dp.feed_update(bot=bot, update=incoming_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:

(continued from previous page)

```
) -> 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
```

aiogram.dispatcher.dispatcher.Dispatcher'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
```

(continues on next page)

```
# 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
) -> Any:
    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+)"))
- 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
```

```
class aiogram.filters.command.Command(*values: str \mid Pattern \mid BotCommand, commands: Sequence[str \mid Pattern \mid BotCommand] \mid str \mid Pattern \mid BotCommand \mid None = None, prefix: <math>str = '/', ignore\_case: bool = False, ignore\_mention: bool = False, magic: MagicFilter \mid None = None)
```

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 | None = None, args: str | None = None, regexp_match: Match[str] | 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	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:** 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**

**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:** 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)

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.

```
class aiogram.filters.exception.ExceptionTypeFilter(*exceptions: Type[Exception])
    Allows to match exception by type
    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.

**Note:** You can use polling from only one polling process per single Bot token, in other case Telegram server will return an error.

Note: If you will need to scale your bot, you should use webhooks instead of long-polling.

**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.)
    mmm
   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!")
```

```
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 Webhook

Telegram Bot API supports webhook. If you set webhook for your bot, Telegram will send updates to the specified url. You can use <code>aiogram.methods.set\_webhook.SetWebhook()</code> method to specify a url and receive incoming updates on it.

**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.

**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 instance
- 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 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) → None
```

Handler that supports multiple bots the context will be resolved from path variable 'bot\_token'

**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.

\_\_init\_\_(ips: Sequence[str | IPv4Network | IPv4Address] | None = None)

So, aiogram has an implementation of the IP filtering middleware for aiohttp.

```
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(\it{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 aiogram import Bot, Dispatcher, Router, types

from aiogram.enums import ParseMode

from aiogram.enums 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: types.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.
\rightarrowetc.)
   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 a default parse mode which will be passed to all API.
\hookrightarrow calls
   bot = Bot(TOKEN, 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.
⇔cases
   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.

```
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, types
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: types.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:
    # 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 a default parse mode which will be passed to all API
\hookrightarrow calls
   bot = Bot(TOKEN, 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...
⇔ cases
    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)
```

Note: 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.

# 2.4.7 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.

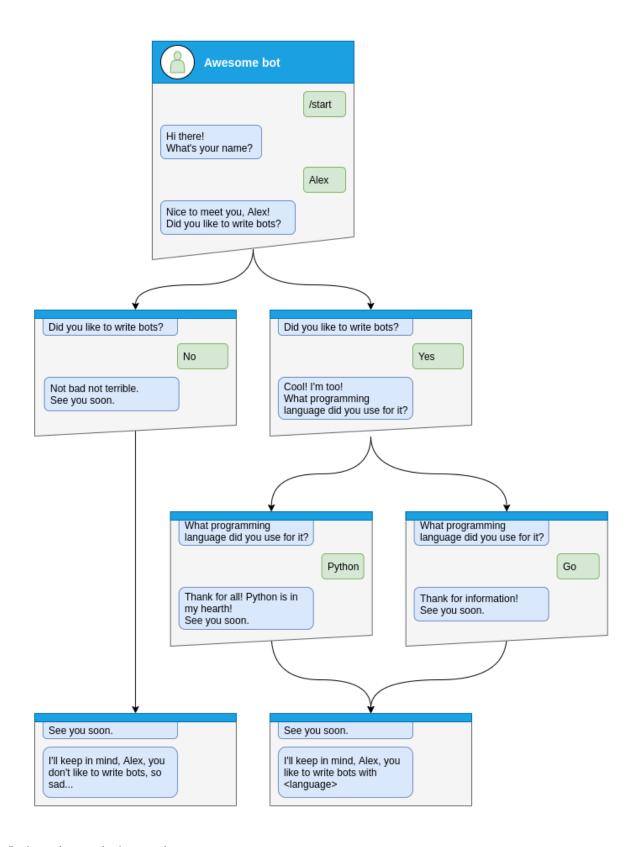
It is an abstract machine that can be in exactly one of a finite number of states at any given time. The FSM can change from one state to another in response to some inputs; the change from one state to another is called a transition.

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

### 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

```
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.enums import ParseMode
   from aiogram.filters import Command, CommandStart
   from aiogram.fsm.context import FSMContext
10
   from aiogram.fsm.state import State, StatesGroup
11
   from aiogram.types import (
12
       KeyboardButton,
13
       Message,
14
       ReplyKeyboardMarkup,
       ReplyKeyboardRemove,
16
   )
18
   TOKEN = getenv("BOT_TOKEN")
20
   form_router = Router()
21
22
23
   class Form(StatesGroup):
24
       name = State()
25
       like_bots = State()
26
       language = State()
27
29
   @form_router.message(CommandStart())
   async def command_start(message: Message, state: FSMContext) -> None:
31
       await state.set_state(Form.name)
       await message.answer(
33
           "Hi there! What's your name?",
           reply_markup=ReplyKeyboardRemove(),
35
       )
37
   @form_router.message(Command("cancel"))
```

```
@form_router.message(F.text.casefold() == "cancel")
40
   async def cancel_handler(message: Message, state: FSMContext) -> None:
42
       Allow user to cancel any action
44
       current_state = await state.get_state()
       if current state is None:
46
           return
48
       logging.info("Cancelling state %r", current_state)
49
       await state.clear()
50
       await message.answer(
51
           "Cancelled.",
           reply_markup=ReplyKeyboardRemove(),
53
       )
55
   @form_router.message(Form.name)
57
   async def process_name(message: Message, state: FSMContext) -> None:
       await state.update_data(name=message.text)
59
       await state.set_state(Form.like_bots)
       await message.answer(
61
           f"Nice to meet you, {html.quote(message.text)}!\nDid you like to write bots?",
           reply_markup=ReplyKeyboardMarkup(
                keyboard=[
                    65
                        KeyboardButton(text="Yes"),
                        KeyboardButton(text="No"),
68
                ],
                resize_keyboard=True,
70
           ),
       )
72
74
   @form_router.message(Form.like_bots, F.text.casefold() == "no")
   async def process_dont_like_write_bots(message: Message, state: FSMContext) -> None:
76
       data = await state.get_data()
       await state.clear()
78
       await message.answer(
           "Not bad not terrible.\nSee you soon.",
80
           reply_markup=ReplyKeyboardRemove(),
81
82
       await show_summary(message=message, data=data, positive=False)
83
84
85
   @form_router.message(Form.like_bots, F.text.casefold() == "yes")
   async def process_like_write_bots(message: Message, state: FSMContext) -> None:
87
       await state.set_state(Form.language)
       await message.reply(
           "Cool! I'm too!\nWhat programming language did you use for it?",
91
```

```
reply_markup=ReplyKeyboardRemove(),
92
        )
    @form_router.message(Form.like_bots)
    async def process_unknown_write_bots(message: Message) -> None:
        await message.reply("I don't understand you :(")
    @form_router.message(Form.language)
101
    async def process_language(message: Message, state: FSMContext) -> None:
102
        data = await state.update_data(language=message.text)
103
        await state.clear()
105
        if message.text.casefold() == "python":
            await message.reply(
107
                "Python, you say? That's the language that makes my circuits light up! "
109
        await show_summary(message=message, data=data)
111
113
    async def show_summary(message: Message, data: Dict[str, Any], positive: bool = True) ->_
114
    →None:
        name = data["name"]
115
        language = data.get("language", "<something unexpected>")
116
        text = f"I'll keep in mind that, {html.quote(name)}, "
117
118
            f"you like to write bots with {html.quote(language)}."
119
            if positive
120
            else "you don't like to write bots, so sad..."
121
        await message.answer(text=text, reply_markup=ReplyKeyboardRemove())
123
124
125
    async def main():
126
        bot = Bot(token=TOKEN, parse_mode=ParseMode.HTML)
127
        dp = Dispatcher()
128
        dp.include_router(form_router)
129
130
        await dp.start_polling(bot)
131
132
133
       __name__ == "__main__":
134
        logging.basicConfig(level=logging.INFO, stream=sys.stdout)
135
        asyncio.run(main())
136
```

### **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: \sim redis.asyncio.client.Redis, key\_builder: \sim aiogram.fsm.storage.base.KeyBuilder \mid None = None, state\_ttl: int \mid \sim datetime.timedelta \mid None = None, data\_ttl: int \mid \sim datetime.timedelta \mid None = None, json\_loads: <math>\sim typing.Callable[[...], \sim typing.Any] = < function loads>, json\_dumps: <math>\sim 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: ~typing.Callable[[...], ~typing.Any] = <function loads>, 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

 $\textbf{classmethod from\_url}(\textit{url: str, connection\_kwargs: Dict[str, Any]} \mid \textit{None} = \textit{None, **kwargs: Any}) \rightarrow \textit{RedisStorage}$ 

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

# 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?
>:<field?>
```

 $\textbf{build}(\textit{key: StorageKey, part: Literal['data', 'state', 'lock']} \mid \textit{None} = \textit{None}) \rightarrow \textit{str}$ 

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]) → 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.)
```

# **Scenes Wizard**

New 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:

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

```
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
```

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

(continues on next page)

```
: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),
    ),
    11.11
    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:
```

(continues on next page)

```
: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():
    dispatcher = create_dispatcher()
    bot = Bot(TOKEN)
    await dispatcher.start_polling(bot)

if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO)
    asyncio.run(main())
    # Alternatively, you can use aiogram-cli:
    # `aiogram run polling quiz_scene:create_dispatcher --log-level info --token BOT_
    →TOKEN`
```

Complete them all

Listing 14: Quiz Example

```
import asyncio
import logging
from dataclasses import dataclass, field
from os import getenv
from typing import Any

(continues on next page)
```

```
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_
→correct)
# Fake data, in real application you should use a database or something else
QUESTIONS = [
    Question(
```

(continues on next page)

```
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".
```

(continues on next page)

```
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
       :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),
       )
   @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>
```

(continues on next page)

```
# 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:
        11 11 11
        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
```

(continues on next page)

```
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():
   dispatcher = create_dispatcher()
   bot = Bot(TOKEN)
    await dispatcher.start_polling(bot)
if __name__ == "__main__":
   logging.basicConfig(level=logging.INFO)
   asyncio.run(main())
   # Alternatively, you can use aiogram-cli:
    # `aiogram run polling quiz_scene:create_dispatcher --log-level info --token BOT_
→ TOKEN`
```

# **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.

**Note:** 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

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.

#### **Parameters**

**kwargs** – Additional keyword arguments passed to the scene's exit method.

# Returns

None

**async enter**( $scene\_type: Type[Scene] \mid str \mid None, \_check\_active: bool = True, **kwargs: Any) <math>\rightarrow$  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

 $\textbf{class} \ \ \text{aiogram.fsm.scene.} \textbf{SceneConfig}(\textit{state: 'Optional[str]', handlers: 'List[HandlerContainer]', actions: 'Dict[SceneAction, Dict[str, CallableObject]]', reset_data_on_enter: 'Optional[bool]' = None, reset_history_on_enter: 'Optional[bool]' = None, callback_query_without_state: 'Optional[bool]' = None)$ 

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.

**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

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] \mid None = None, **kwargs: Any) <math>\rightarrow 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 tree 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

## 2.4.8 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).

**Note:** 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:

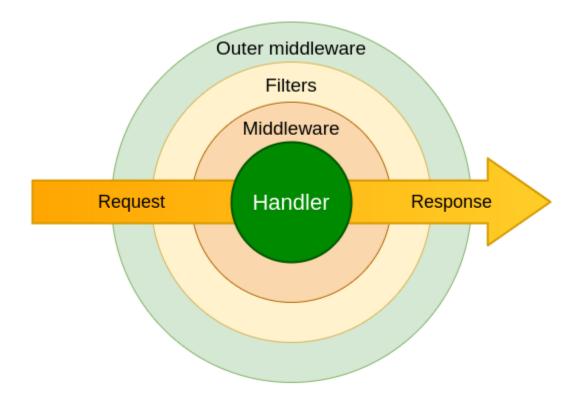
Middleware is reusable software that leverages patterns and frameworks to bridge the gap between the functional requirements of applications and the underlying operating systems, network protocol stacks, and databases.

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(...))



**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

## **2.4.9 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**

## **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** aiogram.exceptions.**TelegramRetryAfter**(*method*: *TelegramMethod*, *message*: *str*, *retry\_after*: *int*)

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.

# $\textbf{exception} \ \ \textbf{aiogram.exceptions.} \textbf{TelegramConflictError} (\textit{method: TelegramMethod, message: str})$

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.)

# 2.4.10 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

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

#### 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: ...
```

```
property from_user: 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:** 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

562

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**

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) → aiogram.utils.keyboard.InlineKeyboardBuilder
```

Add new inline button to markup

```
as_markup() \rightarrow 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 -

2.5. Utils 563

```
• 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() \rightarrow 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
     \textbf{classmethod from\_markup}(\textit{markup}: InlineKeyboardMarkup}) \rightarrow \textit{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[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() \rightarrow 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) → KeyboardBuilder[ButtonType]
     Add row to markup
     When too much buttons is passed it will be separated to many rows
         Parameters

    buttons –

             • width -
```

2.5. Utils 565

#### 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:

**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

**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

 $\_$ \_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

2.5. Utils 567

- 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:

```
class aiogram.utils.i18n.middleware.I18nMiddleware(i18n: 118n, i18n\_key: str \mid None = 'i18n', middleware\_key: str = 'i18n\_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]) → str

Detect current user locale based on event and context.

# This method must be defined in child classes

# **Parameters**

- event -
- data –

#### Returns

 $setup(router: Router, exclude: Set[str] | None = None) \rightarrow BaseMiddleware$ 

Register middleware for all events in the Router

#### **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
```

**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)

2.5. Utils 569

## **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 | int, message_thread_id: int | 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)  $\rightarrow ChatActionSender$ 

Create instance of the sender with record video note action

**classmethod record\_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 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**( $chat\_id$ :  $int \mid str, bot$ :  $Bot, message\_thread\_id$ :  $int \mid None = None, interval$ :  $float = 5.0, initial\_sleep$ : float = 0.0)  $\rightarrow ChatActionSender$ 

Create instance of the sender with upload 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 | 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\_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 ashould 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:
```

(continues on next page)

(continued from previous page)

```
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: \simtyping.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), '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].

This replaces *Model*.\_\_fields\_\_ from Pydantic V1.

```
model_post_init(\_ModelMetaclass\_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),
    '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].

This replaces *Model*.\_\_fields\_\_ from Pydantic V1.

```
model_post_init(\_ModelMetaclass\_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': 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]] = {'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].

This replaces *Model*.\_\_fields\_\_ from Pydantic V1.

```
\textbf{model\_post\_init}(\_\textit{ModelMetaclass\_context}: \textit{Any}) \rightarrow \textit{None}
```

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

### 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"))
```

 $Look\ at\ aiogram.\ utils.\ callback\_answer.\ CallbackAnswer\ Middle ware\ to\ get\ all\ available\ parameters$ 

### 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.

```
@router.callback_query(<filters>)
async def my_handler(query: CallbackQuery, callback_answer: CallbackAnswer):
    if <everything is ok>:
        callback_answer.text = "All is ok"
    else:
        callback_answer.text = "Something wrong"
        callback_answer.cache_time = 10
```

### 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:

### **Description of objects**

class aiogram.utils.callback\_answer.CallbackAnswerMiddleware(pre: bool = False, text:  $str \mid None = None, show\_alert: bool \mid None = None, url: str \mid None = None, url: str \mid None = None, cache\_time: int \mid None = None)$ 

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:  $str \mid None = None$ ,  $show\_alert: bool \mid None = None$ ,  $url: str \mid None = None$ ,  $cache\_time: int \mid None = None$ )  $\rightarrow$  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

 $disable() \rightarrow None$ 

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

property cache_time: int | None
    Response cache time
```

# 2.5.6 Formatting

Make your message formatting flexible and simple

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
aiogram.utils.formatting.as_marked_list(*items: Any, marker: str = '-') \rightarrow 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 -
```

2.5. Utils 581

body –marker –

Returns

aiogram.utils.formatting.as\_numbered\_section(title: Any, \*body: Any, start: int = 1, fmt:  $str = '\{\}.$ ')  $\rightarrow Teyt$ 

Wrap elements as section with numbered list

#### **Parameters**

- title -
- body -
- start -
- fmt -

#### Returns

 $\verb|aiogram.utils.formatting.as_key_value| (\textit{key: Any, value: Any}) \rightarrow \textit{Text}$ 

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),
        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

```
Failed:
                     Test 2
                   Summary:
                                     Total: 4
                                     Success: 3
                                     Failed: 1
                   #test
Or as HTML:
 <b>Success:</b>
   Test 1
   Test 3
   Test 4
<br/>b>Failed:</b>
   Test 2
<b>Summary:</b>
        <br/>tal:</b> 4
        <b>Success:</b> 3
        <br/>b>Failed:</b> 1
#test
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())
```

Test 4

2.5. Utils 583

**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**

```
class aiogram.utils.formatting.Text(*body: Any, **params: Any)
```

Bases: Iterable[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

# 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 | None = None, \*\*kwargs: ~typing.Any) → 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 = & \begin{tabular}{ll} None = & \begin{tabular}{ll} Pone, parse\_mode: str} \mid None = \begin{tabular}{ll} None = None, has\_spoiler: bool} \mid None = None, **kwargs: ~typing.Any) \rightarrow None \\ \end{tabular} None = \begin{tabular}{ll} No
```

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

 $\begin{tabular}{ll} \textbf{add\_video} (\textit{media: str} \mid \sim & aiogram.types.input\_file.InputFile, thumbnail: \sim & aiogram.types.input\_file.InputFile \mid None = None, caption: str \mid None = None, parse\_mode: str \mid None = < Default('parse\_mode') >, caption\_entities: \sim & typing.List[\sim & aiogram.types.message\_entity.MessageEntity] \mid None = None, width: int \mid None = None, height: int \mid None = None, duration: int \mid None = None, supports\_streaming: bool \mid None = None, has\_spoiler: bool \mid None = None, **kwargs: \sim & typing.Any) \rightarrow None \\ \end{tabular}$ 

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:  $Callable[[bytes], bytes] | None = None) <math>\rightarrow str$ 

Decode URL-safe base64url payload with decoder.

# 2.6 Changelog

# 2.6.1 3.7.0 [UNRELEASED DRAFT] (2024-05-10)

### **Features**

• Added new storage aiogram.fsm.storage.MongoStorage for Finite State Machine based on Mongo DB (using motor library) #1434

# 2.6.2 3.6.0 (2024-05-06)

#### **Features**

• Added full support of Bot API 7.3 #1480

### **Improved Documentation**

• Added telegram objects transformation block in  $2.x \rightarrow 3.x$  migration guide #1412

# 2.6.3 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

# 2.6.4 3.4.1 (2024-02-17)

#### **Bugfixes**

• Fixed JSON serialization of the LinkPreviewOptions class while it is passed as bot-wide default options. #1418

# 2.6.5 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.

2.6. Changelog 593

- 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

# 2.6.6 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

# 2.6.7 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. Changelog 595

# 2.6.8 3.1.1 (2023-09-25)

### **Bugfixes**

• Fixed *pydantic* version <2.4, since 2.4 has breaking changes. #1322

# 2.6.9 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

# 2.6.10 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

# 2.6.11 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

# 2.6.12 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

2.6. Changelog 597

# **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

# 2.6.13 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

## 2.6.14 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
)
```

**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

2.6. Changelog 599

### **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.

**Danger: 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

# 2.6.15 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(...)

asyncio.run(main())
```

2.6. Changelog 601

#### **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. aiogram.enums.dice\_emoji.DiceEmoji, content\_type.ContentType, aiogram.enums. inline\_query\_result\_type.InlineQueryResultType, aiogram.enums.input\_media\_type. aiogram.enums.mask\_position\_point.MaskPositionPoint, InputMediaType, 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.
ban(), aiogram.types.chat.Chat.set_description(),
                                                    aiogram.types.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

# 2.6.16 3.0.0b6 (2022-11-18)

#### **Features**

(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

2.6. Changelog 603

# **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

# 2.6.17 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

# 2.6.18 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

2.6. Changelog 605

### **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

# 2.6.19 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

### 2.6.20 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

# 2.6.21 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

```
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)))

@router.message(F.photo[-1].as_("photo"))
async def download_photos_handler(message: Message, photo: PhotoSize, bot: Bot):
    content = await bot.download(photo)
```

#759

2.6. Changelog 607

#### **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

# 2.6.22 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

# 2.6.23 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

### 2.6.24 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

2.6. Changelog 609

• 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

### 2.6.25 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

# 2.6.26 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

### 2.6.27 2.14.3 (2021-07-21)

• Fixed ChatMember type detection via adding customizable object serialization mechanism (#624, #623)

# 2.6.28 2.14.2 (2021-07-26)

- Fixed MemoryStorage cleaner (#619)
- Fixed unused default locale in I18nMiddleware (#562, #563)

### 2.6.29 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)

### 2.6.30 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. Changelog 611

### 2.6.31 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)

# 2.6.32 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)

### 2.6.33 2.11.2 (2021-11-10)

- · Fixed default parse mode
- Added missing "supports\_streaming" argument to answer\_video method #462

### 2.6.34 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)

# 2.6.35 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)

#### 2.6.36 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'

#### 2.6.37 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)

2.6. Changelog 613

- 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)

# 2.6.38 2.9.2 (2021-06-13)

- Fixed Message.get\_full\_command() #352
- Fixed markdown util #353

#### 2.6.39 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)

# 2.6.40 2.8 (2021-04-26)

- Added full support of Bot API 4.8
- Added Message.answer\_dice and Message.reply\_dice methods (#306)

# 2.6.41 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

# 2.6.42 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)

### 2.6.43 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

### 2.6.44 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. Changelog 615

### 2.6.45 2.5.2 (2021-01-01)

• Get back quote\_html and escape\_md functions

### 2.6.46 2.5.1 (2021-01-01)

• Hot-fix of CommandStart filter

### 2.6.47 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...)

# 2.6.48 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

### 2.6.49 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

#### 2.6.50 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)

### 2.6.51 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. Changelog 617

#### 2.6.52 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)

# 2.6.53 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

### 2.6.54 1.4 (2021-08-03)

• Bot API 4.0 (#57)

### 2.6.55 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.

### 2.6.56 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

#### 2.6.57 1.3.1 (2018-05-27)

## 2.6.58 1.3 (2021-04-22)

- Allow to use Socks5 proxy (need manually install aiosocksy).
- Refactored aiogram.utils.executor module.
- [Warning] Updated requirements list.

# 2.6.59 1.2.3 (2018-04-14)

- Fixed API errors detection
- Fixed compability of setup.py with pip 10.0.0

### 2.6.60 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.

### 2.6.61 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.

### 2.6.62 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 documentation
- · Other bug fixes and minor improvements.

2.6. Changelog 619

### 2.6.63 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)

### 2.6.64 1.0.4 (2018-01-10)

### 2.6.65 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.

# 2.6.66 1.0.2 (2017-11-29)

#### 2.6.67 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.

#### 2.6.68 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.

```
2.6.69 0.4.1 (2017-08-03)
```

2.6.70 0.4 (2017-08-05)

2.6.71 0.3.4 (2017-08-04)

2.6.72 0.3.3 (2017-07-05)

2.6.73 0.3.2 (2017-07-04)

2.6.74 0.3.1 (2017-07-04)

2.6.75 0.2b1 (2017-06-00)

2.6.76 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 621

#### **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 623

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**

a	aiogram.methods.close_forum_topic,327
aiogram.dispatcher.flags,556	aiogram.methods.close_general_forum_topic,
aiogram.enums.bot_command_scope_type, 471	328
aiogram.enums.chat_action, 471	aiogram.methods.copy_message,329
aiogram.enums.chat_boost_source_type, 472	aiogram.methods.copy_messages, 331
aiogram.enums.chat_member_status, 472	<pre>aiogram.methods.create_chat_invite_link, 333</pre>
aiogram.enums.chat_type, 473	aiogram.methods.create_forum_topic, 334
aiogram.enums.content_type, 473	aiogram.methods.create_invoice_link, 458
aiogram.enums.currency, 475	aiogram.methods.create_new_sticker_set, 303
aiogram.enums.dice_emoji,478	<pre>aiogram.methods.decline_chat_join_request,</pre>
aiogram.enums.encrypted_passport_element, 478	335
aiogram.enums.inline_query_result_type, 479	<pre>aiogram.methods.delete_chat_photo, 336</pre>
aiogram.enums.input_media_type, 480	<pre>aiogram.methods.delete_chat_sticker_set, 337</pre>
aiogram.enums.keyboard_button_poll_type_type,	aiogram.methods.delete_forum_topic, 338
480	aiogram.methods.delete_message,434
aiogram.enums.mask_position_point, 480	aiogram.methods.delete_messages, 436
aiogram.enums.menu_button_type, 481	aiogram.methods.delete_my_commands, 339
aiogram.enums.message_entity_type, 481	<pre>aiogram.methods.delete_sticker_from_set, 305</pre>
aiogram.enums.message_origin_type, 482	aiogram.methods.delete_sticker_set,306
aiogram.enums.parse_mode, 482	aiogram.methods.delete_webhook,464
aiogram.enums.passport_element_error_type,	<pre>aiogram.methods.edit_chat_invite_link, 340</pre>
482	<pre>aiogram.methods.edit_forum_topic, 342</pre>
aiogram.enums.poll_type, 483	<pre>aiogram.methods.edit_general_forum_topic, 343</pre>
aiogram.enums.reaction_type_type, 483	<pre>aiogram.methods.edit_message_caption, 437</pre>
aiogram.enums.sticker_format, 483	<pre>aiogram.methods.edit_message_live_location,</pre>
aiogram.enums.sticker_type, 483	438
aiogram.enums.topic_icon_color, 484	<pre>aiogram.methods.edit_message_media, 441</pre>
aiogram.enums.update_type, 484	<pre>aiogram.methods.edit_message_reply_markup,</pre>
aiogram.exceptions, 554	442
aiogram.handlers.callback_query,558	<pre>aiogram.methods.edit_message_text, 443</pre>
aiogram.methods.add_sticker_to_set, 302	<pre>aiogram.methods.export_chat_invite_link, 344</pre>
aiogram.methods.answer_callback_query, 321	aiogram.methods.forward_message,345
aiogram.methods.answer_inline_query, 448	aiogram.methods.forward_messages,347
aiogram.methods.answer_pre_checkout_query,	<pre>aiogram.methods.get_business_connection, 348</pre>
456	aiogram.methods.get_chat,349
aiogram.methods.answer_shipping_query, 457	<pre>aiogram.methods.get_chat_administrators, 350</pre>
aiogram.methods.answer_web_app_query, 450	<pre>aiogram.methods.get_chat_member,351</pre>
<pre>aiogram.methods.approve_chat_join_request,</pre>	<pre>aiogram.methods.get_chat_member_count, 352</pre>
322	<pre>aiogram.methods.get_chat_menu_button,353</pre>
aiogram.methods.ban_chat_member, 323	<pre>aiogram.methods.get_custom_emoji_stickers,</pre>
aiogram.methods.ban_chat_sender_chat, 325	307
aiogram.methods.close, 326	<pre>aiogram.methods.get_file,354</pre>
, - · · · · · · · · · · · · · · · · · ·	

```
aiogram.methods.get_forum_topic_icon_stickers,aiogram.methods.set_game_score, 454
                                               aiogram.methods.set_message_reaction, 419
                                               aiogram.methods.set_my_commands, 421
aiogram.methods.get_game_high_scores, 451
aiogram.methods.get_me, 356
                                               aiogram.methods.set_my_default_administrator_rights,
aiogram.methods.get_my_commands,357
aiogram.methods.get_my_default_administrator_raighqsam.methods.set_my_description, 423
                                               aiogram.methods.set_my_name, 424
aiogram.methods.get_my_description, 359
                                               aiogram.methods.set_my_short_description, 425
aiogram.methods.get_my_name, 360
                                               aiogram.methods.set_passport_data_errors,469
aiogram.methods.get_my_short_description, 361
                                               aiogram.methods.set_sticker_emoji_list, 313
aiogram.methods.get_sticker_set, 308
                                               aiogram.methods.set_sticker_keywords, 314
aiogram.methods.get_updates, 465
                                               aiogram.methods.set_sticker_mask_position,
aiogram.methods.get_user_chat_boosts, 361
aiogram.methods.get_user_profile_photos, 362
                                               aiogram.methods.set_sticker_position_in_set,
aiogram.methods.get_webhook_info, 467
aiogram.methods.hide_general_forum_topic, 363
                                               aiogram.methods.set_sticker_set_thumbnail,
aiogram.methods.leave_chat, 364
aiogram.methods.log_out, 365
                                               aiogram.methods.set_sticker_set_title,319
aiogram.methods.pin_chat_message, 366
                                               aiogram.methods.set_webhook, 467
aiogram.methods.promote_chat_member, 367
                                               aiogram.methods.stop_message_live_location,
aiogram.methods.reopen_forum_topic, 370
                                                       445
aiogram.methods.reopen_general_forum_topic,
                                               aiogram.methods.stop_poll,447
        371
                                               aiogram.methods.unban_chat_member, 426
aiogram.methods.replace_sticker_in_set, 309
                                               aiogram.methods.unban_chat_sender_chat, 428
aiogram.methods.restrict_chat_member, 372
                                               aiogram.methods.unhide_general_forum_topic,
aiogram.methods.revoke_chat_invite_link, 373
aiogram.methods.send_animation, 375
                                               aiogram.methods.unpin_all_chat_messages, 430
aiogram.methods.send_audio, 377
                                               aiogram.methods.unpin_all_forum_topic_messages,
aiogram.methods.send_chat_action, 380
aiogram.methods.send_contact, 382
                                               aiogram.methods.unpin_all_general_forum_topic_messages,
aiogram.methods.send_dice, 384
aiogram.methods.send_document, 386
                                               aiogram.methods.unpin_chat_message, 433
aiogram.methods.send_game, 452
                                               aiogram.methods.upload_sticker_file, 320
aiogram.methods.send_invoice, 461
                                               aiogram.types.animation, 18
aiogram.methods.send_location, 388
                                               aiogram.types.audio, 19
aiogram.methods.send_media_group, 391
                                               aiogram.types.background_fill, 20
aiogram.methods.send_message, 393
                                               aiogram.types.background_fill_freeform_gradient,
aiogram.methods.send_photo, 395
aiogram.methods.send_poll,398
                                               aiogram.types.background_fill_gradient, 21
aiogram.methods.send_sticker,310
                                               aiogram.types.background_fill_solid, 21
aiogram.methods.send_venue, 401
                                               aiogram.types.background_type, 22
aiogram.methods.send_video, 403
                                               aiogram.types.background_type_chat_theme, 22
aiogram.methods.send_video_note, 406
                                               aiogram.types.background_type_fill, 23
aiogram.methods.send_voice, 409
                                               aiogram.types.background_type_pattern, 23
aiogram.methods.set_chat_administrator_custom_atindgram.types.background_type_wallpaper, 24
        411
                                               aiogram.types.birthdate, 25
aiogram.methods.set_chat_description, 413
                                               aiogram.types.bot_command, 25
aiogram.methods.set_chat_menu_button, 414
                                               aiogram.types.bot_command_scope, 25
aiogram.methods.set_chat_permissions, 415
                                               aiogram.types.bot_command_scope_all_chat_administrators,
aiogram.methods.set_chat_photo, 416
aiogram.methods.set_chat_sticker_set,417
                                               aiogram.types.bot_command_scope_all_group_chats,
aiogram.methods.set_chat_title, 418
                                                       26
aiogram.methods.set_custom_emoji_sticker_set_tahimnumaniltypes.bot_command_scope_all_private_chats,
        312
                                                       27
```

```
aiogram.types.bot_command_scope_chat, 27
                                               aiogram.types.forum_topic, 132
aiogram.types.bot_command_scope_chat_administramitoggram.types.forum_topic_closed, 132
                                               aiogram.types.forum_topic_created, 133
aiogram.types.bot_command_scope_chat_member,
                                               aiogram.types.forum_topic_edited, 133
                                               aiogram.types.forum_topic_reopened, 134
aiogram.types.bot_command_scope_default, 29
                                               aiogram.types.game, 301
aiogram.types.bot_description, 29
                                               aiogram.types.game_high_score, 302
                                               aiogram.types.general_forum_topic_hidden, 134
aiogram.types.bot_name, 30
aiogram.types.bot_short_description, 30
                                               aiogram.types.general_forum_topic_unhidden,
aiogram.types.business_connection, 30
                                                       134
aiogram.types.business_intro, 31
                                               aiogram.types.giveaway, 134
                                               aiogram.types.giveaway_completed, 135
aiogram.types.business_location, 31
aiogram.types.business_messages_deleted, 32
                                               aiogram.types.giveaway_created, 136
aiogram.types.business_opening_hours, 32
                                               aiogram.types.giveaway_winners, 136
aiogram.types.business_opening_hours_interval,aiogram.types.inaccessible_message, 137
        33
                                               aiogram.types.inline_keyboard_button, 137
aiogram.types.callback_game, 301
                                               aiogram.types.inline_keyboard_markup, 139
aiogram.types.callback_query, 33
                                               aiogram.types.inline_query, 230
aiogram.types.chat, 35
                                               aiogram.types.inline_query_result, 231
aiogram.types.chat_administrator_rights, 50
                                               aiogram.types.inline_query_result_article,
aiogram.types.chat_background, 52
                                                       232
aiogram.types.chat_boost, 52
                                               aiogram.types.inline_query_result_audio, 233
                                               aiogram.types.inline_query_result_cached_audio,
aiogram.types.chat_boost_added, 53
aiogram.types.chat_boost_removed, 53
aiogram.types.chat_boost_source, 54
                                               aiogram.types.inline_query_result_cached_document,
aiogram.types.chat_boost_source_gift_code, 54
aiogram.types.chat_boost_source_giveaway, 54
                                               aiogram.types.inline_query_result_cached_gif,
aiogram.types.chat_boost_source_premium, 55
aiogram.types.chat_boost_updated, 56
                                               aiogram.types.inline_query_result_cached_mpeg4_gif,
aiogram.types.chat_full_info,56
aiogram.types.chat_invite_link,60
                                               aiogram.types.inline_query_result_cached_photo,
aiogram.types.chat_join_request,61
aiogram.types.chat_location, 99
                                               aiogram.types.inline_query_result_cached_sticker,
aiogram.types.chat_member,99
aiogram.types.chat_member_administrator,99
                                               aiogram.types.inline_query_result_cached_video,
aiogram.types.chat_member_banned, 102
aiogram.types.chat_member_left, 102
                                               aiogram.types.inline_query_result_cached_voice,
aiogram.types.chat_member_member, 103
aiogram.types.chat_member_owner, 103
                                               aiogram.types.inline_query_result_contact,
aiogram.types.chat_member_restricted, 104
aiogram.types.chat_member_updated, 105
                                               aiogram.types.inline_query_result_document,
aiogram.types.chat_permissions, 124
aiogram.types.chat_photo, 125
                                               aiogram.types.inline_query_result_game, 255
aiogram.types.chat_shared, 126
                                               aiogram.types.inline_query_result_gif, 256
aiogram.types.chosen_inline_result, 229
                                               aiogram.types.inline_query_result_location,
aiogram.types.contact, 127
aiogram.types.dice, 127
                                               aiogram.types.inline_query_result_mpeg4_gif,
aiogram.types.document, 128
                                                       260
aiogram.types.encrypted_credentials, 281
                                               aiogram.types.inline_query_result_photo, 263
aiogram.types.encrypted_passport_element, 282
                                               aiogram.types.inline_query_result_venue, 265
aiogram.types.error_event,553
                                               aiogram.types.inline_query_result_video, 266
aiogram.types.external_reply_info, 128
                                               aiogram.types.inline_query_result_voice, 268
aiogram.types.file, 131
                                               aiogram.types.inline_query_results_button,
                                                       270
aiogram.types.force_reply, 131
```

```
aiogram.types.input_contact_message_content,
                                               aiogram.types.passport_element_error_data_field,
        271
aiogram.types.input_file, 139
                                               aiogram.types.passport_element_error_file,
aiogram.types.input_invoice_message_content,
                                               aiogram.types.passport_element_error_files,
aiogram.types.input_location_message_content,
       274
                                               aiogram.types.passport_element_error_front_side,
aiogram.types.input_media, 140
aiogram.types.input_media_animation, 140
                                               aiogram.types.passport_element_error_reverse_side,
aiogram.types.input_media_audio, 141
aiogram.types.input_media_document, 142
                                               aiogram.types.passport_element_error_selfie,
aiogram.types.input_media_photo, 144
aiogram.types.input_media_video, 144
                                               aiogram.types.passport_element_error_translation_file,
aiogram.types.input_message_content, 275
                                                       289
aiogram.types.input_poll_option, 146
                                               aiogram.types.passport_element_error_translation_files,
aiogram.types.input_sticker, 277
aiogram.types.input_text_message_content, 275
                                               aiogram.types.passport_element_error_unspecified,
aiogram.types.input_venue_message_content,
                                                       291
                                               aiogram.types.passport_file, 292
aiogram.types.invoice, 292
                                               aiogram.types.photo_size, 211
aiogram.types.keyboard_button, 146
                                               aiogram.types.poll, 211
aiogram.types.keyboard_button_poll_type, 147
                                               aiogram.types.poll_answer, 212
aiogram.types.keyboard_button_request_chat,
                                               aiogram.types.poll_option, 213
                                               aiogram.types.pre_checkout_query, 294
aiogram.types.keyboard_button_request_user,
                                               aiogram.types.proximity_alert_triggered, 213
                                               aiogram.types.reaction_count, 214
aiogram.types.keyboard_button_request_users,
                                               aiogram.types.reaction_type, 214
                                               aiogram.types.reaction_type_custom_emoji, 214
aiogram.types.labeled_price, 293
                                               aiogram.types.reaction_type_emoji, 215
aiogram.types.link_preview_options, 151
                                               aiogram.types.reply_keyboard_markup, 215
aiogram.types.location, 152
                                               aiogram.types.reply_keyboard_remove, 216
aiogram.types.login_url, 153
                                               aiogram.types.reply_parameters, 217
aiogram.types.mask_position, 278
                                               aiogram.types.response_parameters, 218
aiogram.types.maybe_inaccessible_message, 153
                                               aiogram.types.sent_web_app_message, 277
aiogram.types.menu_button, 154
                                               aiogram.types.shared_user, 218
aiogram.types.menu_button_commands, 154
                                               aiogram.types.shipping_address, 295
aiogram.types.menu_button_default, 155
                                               aiogram.types.shipping_option, 295
aiogram.types.menu_button_web_app, 155
                                               aiogram.types.shipping_query, 296
aiogram.types.message, 156
                                               aiogram.types.sticker, 279
aiogram.types.message_auto_delete_timer_changematiogram.types.sticker_set, 280
                                               aiogram.types.story, 219
aiogram.types.message_entity, 205
                                               aiogram.types.successful_payment, 297
aiogram.types.message_id, 206
                                               aiogram.types.switch_inline_query_chosen_chat,
aiogram.types.message_origin, 207
                                                       219
aiogram.types.message_origin_channel, 207
                                               aiogram.types.text_quote, 220
aiogram.types.message_origin_chat, 208
                                               aiogram.types.update, 298
aiogram.types.message_origin_hidden_user, 208
                                               aiogram.types.user, 220
aiogram.types.message_origin_user, 209
                                               aiogram.types.user_chat_boosts, 222
aiogram.types.message_reaction_count_updated,
                                               aiogram.types.user_profile_photos, 222
                                               aiogram.types.user_shared, 223
aiogram.types.message_reaction_updated, 210
                                               aiogram.types.users_shared, 223
aiogram.types.order_info, 293
                                               aiogram.types.venue, 224
aiogram.types.passport_data, 283
                                               aiogram.types.video, 224
                                               aiogram.types.video_chat_ended, 225
aiogram.types.passport_element_error, 283
```

```
aiogram.types.video_chat_participants_invited, 226
aiogram.types.video_chat_scheduled, 226
aiogram.types.video_chat_started, 226
aiogram.types.video_note, 227
aiogram.types.voice, 227
aiogram.types.web_app_data, 228
aiogram.types.web_app_info, 228
aiogram.types.webhook_info, 300
aiogram.types.write_access_allowed, 229
```

# **INDEX**

Symbols	method), 515
call() (aiogram.dispatcher.middlewares.base.BaseM method), 551	liddieware () (aiogram.webhook.security.IPFilter method), 516
call() (aiogram.filters.base.Filter method), 510	A
init() (aiogram.dispatcher.dispatcher.Dispatcher	
method), 495	accent_color_id (aiogram.types.chat.Chat attribute),
init() (aiogram.dispatcher.router.Router method),	35
489	accent_color_id(aiogram.types.chat_full_info.ChatFullInfo
init() (aiogram.filters.command.Command	attribute), 57
method), 500	action (aiogram.methods.send_chat_action.SendChatAction attribute), 380
init() (aiogram.fsm.storage.memory.MemoryStorage	actions (aiogram.fsm.scene.SceneConfig attribute), 547
method), 529	active_usernames (aiogram.types.chat.Chat attribute),
init() (aiogram.fsm.storage.redis.RedisStorage method), 529	36
init() (aiogram.types.input_file.BufferedInputFile	active_usernames(aiogram.types.chat_full_info.ChatFullInfo
method), 488	attribute), 58
init() (aiogram.types.input_file.FSInputFile	actor_chat(aiogram.types.message_reaction_updated.MessageReactionU
method), 487	attribute), 210
init() (aiogram.utils.callback_answer.CallbackAnsw	
method), 579	add() (aiogram.utils.keyboard.InlineKeyboardBuilder
init() (aiogram.utils.callback_answer.CallbackAnsw	add() (aiogram.utils.keyboard.ReplyKeyboardBuilder
method), 579	
init() (aiogram.utils.chat_action.ChatActionSender method), 570	add() (aiogram.utils.media_group.MediaGroupBuilder
init() (aiogram.utils.formatting.Text method), 583	method), 587
init () (aiogram utils i18n middleware Constl18nMi	datidwardio() (aiogram.utils.media_group.MediaGroupBuilder
method), 567	method), 587
init() (aiogram.utils.i18n.middleware.FSMI18nMid	
method), 568	tribute), 52
init () (aiogram.utils.i18n.middleware.I18nMiddlew	andd_document() (aiogram.utils.media_group.MediaGroupBuilder
method), 568	method), 588
init() (aiogram.utils.i18n.middleware.SimpleI18nM	i <b>ddleupho</b> to() (aiogram.utils.media_group.MediaGroupBuilder
method), 567	method), 589
init() (aiogram.utils.keyboard.InlineKeyboardBuild	eadd_to_router() (aiogram.fsm.scene.Scene class
method), 563	method), 545
init() (aiogram.utils.keyboard.ReplyKeyboardBuild	eadd_video() (aiogram.utils.media_group.MediaGroupBuilder
method), 564	method), 589
$\verb \init\()  (a iogram.webhook.aiohttp\_server.BaseReque.$	saddedeto_attachment_menu (aiogram.types.user.User
method), 513	attribute), 221
$\verb \init()  (a iogram.webhook.a iohttp\_server.SimpleRequestion   SimpleRequestion   S$	
method), 514	(aiogram.utils.web_app.WebAppUser at-
init () (ajogram webbook ajohttp server Token Rased	dRequestHaviblute). 5/5

additional_chat_count (aiogram.types.giveaway_winners.GiveawayWinn	
attribute), 136 ADDRESS (aiogram.enums.encrypted_passport_element.Enc	aiogram.enums.message_entity_type crypte <b>n Bdxxbe</b> r4E ement
attribute), 478	aiogram.enums.message_origin_type
address (aiogram.methods.send_venue.SendVenue at-	module, 482
tribute), 402	aiogram.enums.parse_mode
$address ({\it aiogram.types.business\_location.BusinessLocations}) address ({\it aiogram.types.business\_locations}) address ({\it aio$	on module, 482
attribute), 31	<pre>aiogram.enums.passport_element_error_type</pre>
address (aiogram.types.chat_location.ChatLocation at-	module, 482
tribute), 99	aiogram.enums.poll_type
address (aiogram.types.inline_query_result_venue.Inline(	
attribute), 265 address (aiogram.types.input_venue_message_content.Inp	aiogram.enums.reaction_type_type
attribute), 276	aiogram.enums.sticker_format
address (aiogram.types.venue.Venue attribute), 224	module, 483
	aiogram.enums.sticker_type
aiogram.methods.add_sticker_to_set), 302	module, 483
<pre>adjust() (aiogram.utils.keyboard.InlineKeyboardBuilder</pre>	
method), 563	module, 484
$\verb"adjust()" (a iogram.utils.keyboard.ReplyKeyboardBuilder")"$	aiogram.enums.update_type
method), 565	module, 484
ADMINISTRATOR (aiogram.enums.chat_member_status.Cha	
attribute), 472	module, 554
AED (aiogram.enums.currency.Currency attribute), 475	aiogram.handlers.callback_query
AFN (aiogram.enums.currency.Currency attribute), 475	module, 558
aiogram.dispatcher.flags module,556	<pre>aiogram.methods.add_sticker_to_set   module, 302</pre>
aiogram.enums.bot_command_scope_type	aiogram.methods.answer_callback_query
module, 471	module, 321
aiogram.enums.chat_action	aiogram.methods.answer_inline_query
module, 471	module, 448
aiogram.enums.chat_boost_source_type	<pre>aiogram.methods.answer_pre_checkout_query</pre>
module, 472	module, 456
aiogram.enums.chat_member_status module,472	<pre>aiogram.methods.answer_shipping_query module, 457</pre>
aiogram.enums.chat_type	aiogram.methods.answer_web_app_query
module, 473	module, 450
<pre>aiogram.enums.content_type   module, 473</pre>	<pre>aiogram.methods.approve_chat_join_request   module, 322</pre>
aiogram.enums.currency	aiogram.methods.ban_chat_member
module, 475	module, 323
aiogram.enums.dice_emoji module,478	<pre>aiogram.methods.ban_chat_sender_chat   module, 325</pre>
aiogram.enums.encrypted_passport_element	aiogram.methods.close
module, 478	module, 326
aiogram.enums.inline_query_result_type module,479	<pre>aiogram.methods.close_forum_topic module,327</pre>
aiogram.enums.input_media_type	<pre>aiogram.methods.close_general_forum_topic</pre>
module, 480	module, 328
aiogram.enums.keyboard_button_poll_type_type module,480	<pre>aiogram.methods.copy_message   module, 329</pre>
aiogram.enums.mask_position_point module,480	aiogram.methods.copy_messages module,331

```
aiogram.methods.create_chat_invite_link
                                               aiogram.methods.get_chat_administrators
   module, 333
                                                   module, 350
aiogram.methods.create_forum_topic
                                               aiogram.methods.get_chat_member
   module, 334
                                                   module, 351
aiogram.methods.create_invoice_link
                                               aiogram.methods.get_chat_member_count
   module, 458
                                                   module, 352
aiogram.methods.create_new_sticker_set
                                               aiogram.methods.get_chat_menu_button
    module, 303
                                                   module, 353
aiogram.methods.decline_chat_join_request
                                               aiogram.methods.get_custom_emoji_stickers
    module, 335
                                                   module, 307
aiogram.methods.delete_chat_photo
                                               aiogram.methods.get_file
    module, 336
                                                   module, 354
aiogram.methods.delete_chat_sticker_set
                                               aiogram.methods.get_forum_topic_icon_stickers
                                                   module, 355
    module, 337
aiogram.methods.delete_forum_topic
                                               aiogram.methods.get_game_high_scores
    module, 338
                                                   module, 451
aiogram.methods.delete_message
                                               aiogram.methods.get_me
    module, 434
                                                   module, 356
                                               aiogram.methods.get_my_commands
aiogram.methods.delete_messages
    module, 436
                                                   module, 357
aiogram.methods.delete_my_commands
                                               aiogram.methods.get_my_default_administrator_rights
    module, 339
                                                   module, 358
aiogram.methods.delete_sticker_from_set
                                               aiogram.methods.get_my_description
    module, 305
                                                   module, 359
                                               aiogram.methods.get_my_name
aiogram.methods.delete_sticker_set
   module, 306
                                                   module, 360
aiogram.methods.delete_webhook
                                               aiogram.methods.get_my_short_description
    module, 464
                                                   module, 361
aiogram.methods.edit_chat_invite_link
                                               aiogram.methods.get_sticker_set
    module, 340
                                                   module, 308
                                               aiogram.methods.get_updates
aiogram.methods.edit_forum_topic
    module, 342
                                                   module, 465
aiogram.methods.edit_general_forum_topic
                                               aiogram.methods.get_user_chat_boosts
   module, 343
                                                   module, 361
aiogram.methods.edit_message_caption
                                               aiogram.methods.get_user_profile_photos
   module, 437
                                                   module, 362
aiogram.methods.edit_message_live_location
                                               aiogram.methods.get_webhook_info
    module, 438
                                                   module, 467
aiogram.methods.edit_message_media
                                               aiogram.methods.hide_general_forum_topic
   module, 441
                                                   module, 363
aiogram.methods.edit_message_reply_markup
                                               aiogram.methods.leave_chat
    module, 442
                                                   module, 364
aiogram.methods.edit_message_text
                                               aiogram.methods.log_out
    module, 443
                                                   module, 365
aiogram.methods.export_chat_invite_link
                                               aiogram.methods.pin_chat_message
    module, 344
                                                   module, 366
aiogram.methods.forward_message
                                               aiogram.methods.promote_chat_member
    module, 345
                                                   module, 367
aiogram.methods.forward_messages
                                               aiogram.methods.reopen_forum_topic
    module, 347
                                                   module, 370
aiogram.methods.get_business_connection
                                               aiogram.methods.reopen_general_forum_topic
    module, 348
                                                   module, 371
aiogram.methods.get_chat
                                               aiogram.methods.replace_sticker_in_set
    module, 349
                                                   module, 309
```

<pre>aiogram.methods.restrict_chat_member module, 372</pre>	<pre>aiogram.methods.set_custom_emoji_sticker_set_thumbnail    module, 312</pre>
<pre>aiogram.methods.revoke_chat_invite_link   module, 373</pre>	<pre>aiogram.methods.set_game_score   module, 454</pre>
aiogram.methods.send_animation module, 375	<pre>aiogram.methods.set_message_reaction   module, 419</pre>
<pre>aiogram.methods.send_audio   module, 377</pre>	<pre>aiogram.methods.set_my_commands     module, 421</pre>
aiogram.methods.send_chat_action module, 380	<pre>aiogram.methods.set_my_default_administrator_rights   module, 422</pre>
aiogram.methods.send_contact module, 382	<pre>aiogram.methods.set_my_description   module, 423</pre>
<pre>aiogram.methods.send_dice   module, 384</pre>	<pre>aiogram.methods.set_my_name   module, 424</pre>
<pre>aiogram.methods.send_document   module, 386</pre>	<pre>aiogram.methods.set_my_short_description   module, 425</pre>
<pre>aiogram.methods.send_game   module, 452</pre>	<pre>aiogram.methods.set_passport_data_errors module, 469</pre>
<pre>aiogram.methods.send_invoice   module, 461</pre>	<pre>aiogram.methods.set_sticker_emoji_list   module, 313</pre>
aiogram.methods.send_location module, 388	<pre>aiogram.methods.set_sticker_keywords   module, 314</pre>
<pre>aiogram.methods.send_media_group   module, 391</pre>	<pre>aiogram.methods.set_sticker_mask_position   module, 315</pre>
<pre>aiogram.methods.send_message   module, 393</pre>	<pre>aiogram.methods.set_sticker_position_in_set   module, 316</pre>
<pre>aiogram.methods.send_photo   module, 395</pre>	<pre>aiogram.methods.set_sticker_set_thumbnail    module, 317</pre>
aiogram.methods.send_poll module,398	<pre>aiogram.methods.set_sticker_set_title   module, 319</pre>
<pre>aiogram.methods.send_sticker module,310</pre>	<pre>aiogram.methods.set_webhook   module, 467</pre>
aiogram.methods.send_venue module, 401	<pre>aiogram.methods.stop_message_live_location   module, 445</pre>
aiogram.methods.send_video module, 403	<pre>aiogram.methods.stop_poll    module, 447</pre>
aiogram.methods.send_video_note module, 406	<pre>aiogram.methods.unban_chat_member module, 426</pre>
aiogram.methods.send_voice module, 409	<pre>aiogram.methods.unban_chat_sender_chat   module, 428</pre>
	stom_atiotgleam.methods.unhide_general_forum_topic module,429
aiogram.methods.set_chat_description module, 413	<pre>aiogram.methods.unpin_all_chat_messages   module, 430</pre>
aiogram.methods.set_chat_menu_button module,414	<pre>aiogram.methods.unpin_all_forum_topic_messages module, 431</pre>
aiogram.methods.set_chat_permissions module, 415	<pre>aiogram.methods.unpin_all_general_forum_topic_messages module, 432</pre>
aiogram.methods.set_chat_photo module,416	aiogram.methods.unpin_chat_message module, 433
aiogram.methods.set_chat_sticker_set module,417	aiogram.methods.upload_sticker_file module, 320
aiogram.methods.set_chat_title module,418	aiogram.types.animation module, 18

```
aiogram.types.audio
                                                aiogram.types.business_opening_hours
   module, 19
                                                    module, 32
aiogram.types.background_fill
                                                aiogram.types.business_opening_hours_interval
   module, 20
                                                    module, 33
aiogram.types.background_fill_freeform_gradientiogram.types.callback_game
                                                    module, 301
   module, 20
aiogram.types.background_fill_gradient
                                                aiogram.types.callback_query
    module, 21
                                                    module, 33
aiogram.types.background_fill_solid
                                                aiogram.types.chat
    module, 21
                                                    module, 35
aiogram.types.background_type
                                                aiogram.types.chat_administrator_rights
    module, 22
                                                    module, 50
aiogram.types.background_type_chat_theme
                                                aiogram.types.chat_background
    module, 22
                                                    module, 52
                                                aiogram.types.chat_boost
aiogram.types.background_type_fill
    module, 23
                                                    module, 52
aiogram.types.background_type_pattern
                                                aiogram.types.chat_boost_added
                                                    module, 53
    module, 23
aiogram.types.background_type_wallpaper
                                                aiogram.types.chat_boost_removed
    module, 24
                                                    module, 53
aiogram.types.birthdate
                                                aiogram.types.chat_boost_source
    module, 25
                                                    module, 54
aiogram.types.bot_command
                                                aiogram.types.chat_boost_source_gift_code
    module, 25
                                                    module, 54
aiogram.types.bot_command_scope
                                                aiogram.types.chat_boost_source_giveaway
    module, 25
                                                    module, 54
aiogram.types.bot_command_scope_all_chat_adminaistgratmortsypes.chat_boost_source_premium
    module, 26
                                                    module, 55
aiogram.types.bot_command_scope_all_group_chataiogram.types.chat_boost_updated
    module, 26
                                                    module, 56
aiogram.types.bot_command_scope_all_private_chanitsgram.types.chat_full_info
    module, 27
                                                    module, 56
aiogram.types.bot_command_scope_chat
                                                aiogram.types.chat_invite_link
    module, 27
                                                    module, 60
aiogram.types.bot_command_scope_chat_administrantogram.types.chat_join_request
   module, 28
                                                    module, 61
aiogram.types.bot_command_scope_chat_member
                                                aiogram.types.chat_location
    module, 28
                                                    module, 99
aiogram.types.bot_command_scope_default
                                                aiogram.types.chat_member
                                                    module, 99
   module, 29
aiogram.types.bot_description
                                                aiogram.types.chat_member_administrator
    module, 29
                                                    module, 99
aiogram.types.bot_name
                                                aiogram.types.chat_member_banned
    module, 30
                                                    module, 102
                                                aiogram.types.chat_member_left
aiogram.types.bot_short_description
                                                    module, 102
    module, 30
aiogram.types.business_connection
                                                aiogram.types.chat_member_member
    module, 30
                                                    module, 103
aiogram.types.business_intro
                                                aiogram.types.chat_member_owner
    module, 31
                                                    module, 103
aiogram.types.business_location
                                                aiogram.types.chat_member_restricted
    module, 31
                                                    module, 104
aiogram.types.business_messages_deleted
                                                aiogram.types.chat_member_updated
    module, 32
                                                    module, 105
```

aiogram.types.chat_permissions module,124	<pre>aiogram.types.inline_keyboard_button module, 137</pre>
aiogram.types.chat_photo module,125	aiogram.types.inline_keyboard_markup module,139
aiogram.types.chat_shared module,126	aiogram.types.inline_query module,230
aiogram.types.chosen_inline_result module,229	aiogram.types.inline_query_result module,231
aiogram.types.contact module,127	<pre>aiogram.types.inline_query_result_article   module, 232</pre>
aiogram.types.dice module,127	<pre>aiogram.types.inline_query_result_audio   module, 233</pre>
aiogram.types.document module,128	<pre>aiogram.types.inline_query_result_cached_audio   module, 235</pre>
aiogram.types.encrypted_credentials module,281	<pre>aiogram.types.inline_query_result_cached_document   module, 237</pre>
aiogram.types.encrypted_passport_element module,282	<pre>aiogram.types.inline_query_result_cached_gif module, 239</pre>
aiogram.types.error_event module,553	<pre>aiogram.types.inline_query_result_cached_mpeg4_gif module, 241</pre>
aiogram.types.external_reply_info module,128	<pre>aiogram.types.inline_query_result_cached_photo   module, 243</pre>
aiogram.types.file module,131	aiogram.types.inline_query_result_cached_sticker module, 245
aiogram.types.force_reply module, 131	aiogram.types.inline_query_result_cached_video module,247
aiogram.types.forum_topic module,132	aiogram.types.inline_query_result_cached_voice module, 250
<pre>aiogram.types.forum_topic_closed   module, 132</pre>	<pre>aiogram.types.inline_query_result_contact   module, 252</pre>
<pre>aiogram.types.forum_topic_created   module, 133</pre>	<pre>aiogram.types.inline_query_result_document   module, 253</pre>
<pre>aiogram.types.forum_topic_edited   module, 133</pre>	<pre>aiogram.types.inline_query_result_game   module, 255</pre>
<pre>aiogram.types.forum_topic_reopened   module, 134</pre>	<pre>aiogram.types.inline_query_result_gif   module, 256</pre>
aiogram.types.game module,301	<pre>aiogram.types.inline_query_result_location module, 258</pre>
aiogram.types.game_high_score module,302	<pre>aiogram.types.inline_query_result_mpeg4_gif module, 260</pre>
aiogram.types.general_forum_topic_hidden module,134	<pre>aiogram.types.inline_query_result_photo   module, 263</pre>
<pre>aiogram.types.general_forum_topic_unhidden   module, 134</pre>	<pre>aiogram.types.inline_query_result_venue   module, 265</pre>
aiogram.types.giveaway module,134	<pre>aiogram.types.inline_query_result_video   module, 266</pre>
<pre>aiogram.types.giveaway_completed   module, 135</pre>	aiogram.types.inline_query_result_voice module,268
aiogram.types.giveaway_created module, 136	aiogram.types.inline_query_results_button module,270
aiogram.types.giveaway_winners module,136	<pre>aiogram.types.input_contact_message_content   module, 271</pre>
aiogram.types.inaccessible_message module,137	<pre>aiogram.types.input_file   module, 139</pre>

<pre>aiogram.types.input_invoice_message_content    module, 271</pre>	<pre>aiogram.types.menu_button_default   module, 155</pre>
<pre>aiogram.types.input_location_message_content    module, 274</pre>	<pre>aiogram.types.menu_button_web_app module, 155</pre>
<pre>aiogram.types.input_media   module, 140</pre>	aiogram.types.message module,156
<pre>aiogram.types.input_media_animation   module, 140</pre>	<pre>aiogram.types.message_auto_delete_timer_changed   module, 205</pre>
<pre>aiogram.types.input_media_audio   module, 141</pre>	aiogram.types.message_entity module, 205
<pre>aiogram.types.input_media_document   module, 142</pre>	aiogram.types.message_id module,206
<pre>aiogram.types.input_media_photo   module, 144</pre>	aiogram.types.message_origin module,207
<pre>aiogram.types.input_media_video   module, 144</pre>	aiogram.types.message_origin_channel module, 207
<pre>aiogram.types.input_message_content   module, 275</pre>	aiogram.types.message_origin_chat module,208
aiogram.types.input_poll_option module, 146	aiogram.types.message_origin_hidden_user module, 208
<pre>aiogram.types.input_sticker module, 277</pre>	aiogram.types.message_origin_user module,209
<pre>aiogram.types.input_text_message_content    module, 275</pre>	aiogram.types.message_reaction_count_updated module, 209
<pre>aiogram.types.input_venue_message_content    module, 276</pre>	<pre>aiogram.types.message_reaction_updated   module, 210</pre>
aiogram.types.invoice module, 292	aiogram.types.order_info module,293
aiogram.types.keyboard_button module, 146	aiogram.types.passport_data module, 283
<pre>aiogram.types.keyboard_button_poll_type   module, 147</pre>	<pre>aiogram.types.passport_element_error   module, 283</pre>
<pre>aiogram.types.keyboard_button_request_chat   module, 148</pre>	<pre>aiogram.types.passport_element_error_data_field   module, 284</pre>
<pre>aiogram.types.keyboard_button_request_user   module, 149</pre>	<pre>aiogram.types.passport_element_error_file   module, 285</pre>
<pre>aiogram.types.keyboard_button_request_users   module, 150</pre>	<pre>aiogram.types.passport_element_error_files   module, 286</pre>
<pre>aiogram.types.labeled_price   module, 293</pre>	<pre>aiogram.types.passport_element_error_front_side   module, 287</pre>
<pre>aiogram.types.link_preview_options   module, 151</pre>	<pre>aiogram.types.passport_element_error_reverse_side   module, 287</pre>
aiogram.types.location module, 152	<pre>aiogram.types.passport_element_error_selfie   module, 288</pre>
<pre>aiogram.types.login_url   module, 153</pre>	<pre>aiogram.types.passport_element_error_translation_file module, 289</pre>
<pre>aiogram.types.mask_position   module, 278</pre>	${\tt aiogram.types.passport\_element\_error\_translation\_files} \\ {\tt module, 290}$
<pre>aiogram.types.maybe_inaccessible_message   module, 153</pre>	<pre>aiogram.types.passport_element_error_unspecified   module, 291</pre>
<pre>aiogram.types.menu_button   module, 154</pre>	<pre>aiogram.types.passport_file   module, 292</pre>
aiogram.types.menu_button_commands module, 154	aiogram.types.photo_size module,211

aiogram.types.poll module, 211	<pre>aiogram.types.user_profile_photos   module, 222</pre>
<pre>aiogram.types.poll_answer module, 212</pre>	aiogram.types.user_shared module, 223
<pre>aiogram.types.poll_option   module, 213</pre>	aiogram.types.users_shared module, 223
aiogram.types.pre_checkout_query module, 294	aiogram.types.venue module, 224
aiogram.types.proximity_alert_triggered module, 213	aiogram.types.video module, 224
aiogram.types.reaction_count module, 214	aiogram.types.video_chat_ended module, 225
aiogram.types.reaction_type module, 214	aiogram.types.video_chat_participants_invited module, 226
aiogram.types.reaction_type_custom_emoji module, 214	aiogram.types.video_chat_scheduled module, 226
aiogram.types.reaction_type_emoji module, 215	aiogram.types.video_chat_started module, 226
aiogram.types.reply_keyboard_markup module, 215	aiogram.types.video_note module, 227
aiogram.types.reply_keyboard_remove module, 216	aiogram.types.voice module, 227
aiogram.types.reply_parameters module, 217	aiogram.types.web_app_data module, 228
aiogram.types.response_parameters module, 218	aiogram.types.web_app_info module, 228
aiogram.types.sent_web_app_message module,277	aiogram.types.webhook_info module, 300
aiogram.types.shared_user module, 218	aiogram.types.write_access_allowed module, 229
aiogram.types.shipping_address module, 295	AiogramError, 554 AiohttpSession (class in
aiogram.types.shipping_option	aiogram.client.session.aiohttp), 15
module, 295 aiogram.types.shipping_query	ALL_(aiogram.enums.currency.Currency attribute), 475 ALL_CHAT_ADMINISTRATORS
module, 296	(aiogram.enums.bot_command_scope_type.BotCommandScope
aiogram.types.sticker module, 279	attribute), 471  ALL_GROUP_CHATS (aiogram.enums.bot_command_scope_type.BotComm
aiogram.types.sticker_set module, 280	attribute), 471  ALL_PRIVATE_CHATS (aiogram.enums.bot_command_scope_type.BotCom
aiogram.types.story module, 219	attribute), 471 allow_bot_chats (aiogram.types.switch_inline_query_chosen_chat.Swi
aiogram.types.successful_payment module, 297	attribute), 220 allow_channel_chats
<pre>aiogram.types.switch_inline_query_chosen_chat module, 219</pre>	(aiogram.types.switch_inline_query_chosen_chat.SwitchInlineQattribute), 220
<pre>aiogram.types.text_quote   module, 220</pre>	allow_group_chats (aiogram.types.switch_inline_query_chosen_chat.S attribute), 220
aiogram.types.update module, 298	allow_sending_without_reply (aiogram.methods.copy_message.CopyMessage
aiogram.types.user	attribute), 330
module, 220	allow_sending_without_reply
<pre>aiogram.types.user_chat_boosts module, 222</pre>	(aiogram.methods.send_animation.SendAnimation attribute), 376

allow_sending_without_reply	attribute), 466
(aiogram.methods.send_audio.SendAudio	allowed_updates (aiogram.methods.set_webhook.SetWebhook
attribute), 379	attribute), 468
allow_sending_without_reply (aiogram.methods.send_contact.SendContact	allowed_updates (aiogram.types.webhook_info.WebhookInfo attribute), 301
attribute), 383	allows_multiple_answers
allow_sending_without_reply	(aiogram.methods.send_poll.SendPoll at-
(aiogram.methods.send_dice.SendDice at-	tribute), 399
<i>tribute</i> ), 385	allows_multiple_answers (aiogram.types.poll.Poll
allow_sending_without_reply	attribute), 212
(aiogram.methods.send_document.SendDocument	tallows_write_to_pm(aiogram.utils.web_app.WebAppUser
attribute), 387	attribute), 575
allow_sending_without_reply	AMD (aiogram.enums.currency.Currency attribute), 475
(aiogram.methods.send_game.SendGame	amount (aiogram.types.labeled_price.LabeledPrice at-
attribute), 453	tribute), 293
allow_sending_without_reply (aiogram.methods.send_invoice.SendInvoice	ANIMATED (aiogram.enums.sticker_format.StickerFormat attribute), 483
attribute), 463	ANIMATION (aiogram.enums.content_type.ContentType
allow_sending_without_reply	attribute), 473
(aiogram.methods.send_location.SendLocation	ANIMATION (aiogram.enums.input_media_type.InputMediaType
attribute), 390	attribute), 480
allow_sending_without_reply	animation(aiogram.methods.send_animation.SendAnimation
(aiogram.methods.send_media_group.SendMedia	aGroup attribute), 375
attribute), 392	${\tt animation} (a iogram. types. external\_reply\_info. ExternalReplyInfo$
allow_sending_without_reply	attribute), 129
(aiogram.methods.send_message.SendMessage	animation (aiogram.types.game.Game attribute), 302
attribute), 394	animation (aiogram.types.message.Message attribute), 159
allow_sending_without_reply (aiogram.methods.send_photo.SendPhoto	Animation (class in aiogram.types.animation), 18
attribute), 397	answer() (aiogram.types.callback_query.CallbackQuery
allow_sending_without_reply	method), 34
(aiogram.methods.send_poll.SendPoll at-	answer() (aiogram.types.chat_join_request.ChatJoinRequest
tribute), 400	method), 62
allow_sending_without_reply	answer() (aiogram.types.chat_member_updated.ChatMemberUpdated
$(a iogram. methods. send\_sticker. Send Sticker$	method), 106
attribute), 311	answer() (aiogram.types.inline_query.InlineQuery
allow_sending_without_reply	method), 230
(aiogram.methods.send_venue.SendVenue	answer() (aiogram.types.message.Message method),
<pre>attribute), 402 allow_sending_without_reply</pre>	answer() (aiogram.types.pre_checkout_query.PreCheckoutQuery
(aiogram.methods.send_video.SendVideo	method), 294
attribute), 405	answer() (aiogram.types.shipping_query.ShippingQuery
allow_sending_without_reply	method), 296
	otanswer_animation() (aiogram.types.chat_join_request.ChatJoinRequest
attribute), 408	method), 64
allow_sending_without_reply	$\verb"answer_animation" (a iogram. types. chat\_member\_up dated. Chat Member\_up dated. Chat$
(aiogram.methods.send_voice.SendVoice	method), 107
attribute), 410	answer_animation() (aiogram.types.message.Message
allow_sending_without_reply	method), 164
(aiogram.types.reply_parameters.ReplyParameteattribute), 217	ranswer_animation_pm() (aiogram.types.chat_join_request.ChatJoinRequest
allow_user_chats (aiogram.types.switch_inline_query_t	
attribute), 219	answer_audio() (aiogram.types.chat_join_request.ChatJoinRequest
allowed_updates(aiogram.methods.get updates.GetUpd	

```
answer_audio() (aiogram.types.chat_member_updated.ChatMembenlethdod)d177
                             method), 108
                                                                                                                                                                                answer_location_pm()
answer_audio()
                                                                      (aiogram.types.message.Message
                                                                                                                                                                                                              (aiogram.types.chat join request.ChatJoinRequest
                                                                                                                                                                                                              method), 79
                            method), 167
answer_audio_pm() (aiogram.types.chat_join_request.Chardswellequedia_group()
                            method), 67
                                                                                                                                                                                                              (aiogram.types.chat join request.ChatJoinRequest
answer_contact() (aiogram.types.chat join request.ChatJoinRequesethod), 80
                             method), 69
                                                                                                                                                                                answer_media_group()
answer_contact() (aiogram.types.chat_member_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated.ChatMember_updated
                            method), 109
                                                                                                                                                                                                             method), 115
answer_contact()
                                                                      (aiogram.types.message.Message answer_media_group()
                            method), 169
                                                                                                                                                                                                              (aiogram.types.message.Message
                                                                                                                                                                                                                                                                                                                               method),
answer_contact_pm()
                            (aiogram.types.chat_join_request.ChatJoinRequesanswer_media_group_pm()
                                                                                                                                                                                                              (aiogram.types.chat_join_request.ChatJoinRequest
                            method), 70
answer_dice() (aiogram.types.chat_join_request.ChatJoinRequest_method), 80
                                                                                                                                                                                answer_photo() (aiogram.types.chat_join_request.ChatJoinRequest
                             method), 86
answer_dice() (aiogram.types.chat_member_updated.ChatMember byedlaveld, 81
                                                                                                                                                                                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.ChatMemberU
                            method), 118
                                                                      (aiogram.types.message.Message
answer_dice()
                                                                                                                                                                                                              method), 115
                            method), 187
                                                                                                                                                                                answer_photo()
                                                                                                                                                                                                                                                       (aiogram.types.message.Message
answer_dice_pm() (aiogram.types.chat_join_request.ChatJoinRequesethod), 182
                             method), 87
                                                                                                                                                                                answer_photo_pm() (aiogram.types.chat_join_request.ChatJoinRequest
answer_document() (aiogram.types.chat join request.ChatJoinRequesthod), 82
                                                                                                                                                                                answer_pm() (aiogram.types.chat_join_request.ChatJoinRequest
                            method), 70
answer_document() (aiogram.types.chat_member_updated.ChatMemberbbb)dated
                             method), 110
                                                                                                                                                                                answer_poll() (aiogram.types.chat_join_request.ChatJoinRequest
answer_document() (aiogram.types.message.Message
                                                                                                                                                                                                              method), 83
                            method), 171
                                                                                                                                                                                answer_poll() (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
answer_document_pm()
                                                                                                                                                                                                              method), 117
                             (aiogram.types.chat_join_request.ChatJoinRequesanswer_poll()
                                                                                                                                                                                                                                                       (aiogram.types.message.Message
                            method), 72
                                                                                                                                                                                                              method), 185
answer_game() (aiogram.types.chat_join_request.ChatJoinRequest_poll_pm() (aiogram.types.chat_join_request.ChatJoinRequest
                            method), 73
                                                                                                                                                                                                              method), 85
answer_game() (aiogram.types.chat_member_updated.ChatMsmebres&pidited() (aiogram.types.chat_join_request.ChatJoinRequest
                            method), 111
                                                                                                                                                                                                              method), 88
answer_game()
                                                                       (aiogram.types.message.Message answer_sticker() (aiogram.types.chat member updated.ChatMemberU
                             method), 172
                                                                                                                                                                                                              method), 119
answer_game_pm() (aiogram.types.chat_join_request.Chataniswer_particker()
                                                                                                                                                                                                                                                      (aiogram.types.message.Message
                            method), 74
                                                                                                                                                                                                             method), 188
answer_invoice() (aiogram.types.chat join request.Chatahaswer_invoice() (aiogram.types.chat join request.Chatahaswer_invoice()
                             method), 74
                                                                                                                                                                                                              (aiogram.types.chat_join_request.ChatJoinRequest
answer_invoice() (aiogram.types.chat member updated.ChatMembæthbpblated
                                                                                                                                                                                answer_venue() (aiogram.types.chat_join_request.ChatJoinRequest
                            method), 112
answer_invoice()
                                                                                                                                                                                                              method), 89
                                                                     (aiogram.types.message.Message
                             method), 175
                                                                                                                                                                                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_invoice_pm()
                                                                                                                                                                                                              method), 120
                             (aiogram.types.chat_join_request.ChatJoinRequesanswer_venue()
                                                                                                                                                                                                                                                       (aiogram.types.message.Message
                            method), 76
                                                                                                                                                                                                              method), 190
answer_location() (aiogram.types.chat_join_request.Chatlowertequexte_pm() (aiogram.types.chat_join_request.ChatloinRequest
                                                                                                                                                                                                              method), 90
                             method), 78
answer_location() (aiogram.types.chat member updatedxiSwatMexiblerd(p)datachram.types.chat join request.ChatJoinRequest
                             method), 114
                                                                                                                                                                                                              method), 91
```

answer\_location() (aiogram.types.message.Message answer\_video() (aiogram.types.chat member updated.ChatMemberUpd

T 10 404	
method), 121	ARTICLE (aiogram.enums.inline_query_result_type.InlineQueryResultTyp
answer_video() (aiogram.types.message.Message	attribute), 479
<pre>method), 192 answer_video_note()</pre>	as_handler() (aiogram.fsm.scene.Scene class method), 545
	esats_html() (aiogram.utils.formatting.Text method), 584
method), 94	as_key_value() (in module aiogram.utils.formatting),
answer_video_note()	582
(aiogram.types.chat_member_updated.ChatMem	
method), 122	583
<pre>answer_video_note()</pre>	as_line() (in module aiogram.utils.formatting), 580
(aiogram.types.message.Message method),	as_list() (in module aiogram.utils.formatting), 581
194	<pre>as_markdown() (aiogram.utils.formatting.Text method),</pre>
<pre>answer_video_note_pm()</pre>	584
(aiogram.types.chat_join_request.ChatJoinReque	
method), 95	aiogram.utils.formatting), 581
answer_video_pm() (aiogram.types.chat_join_request.Cl	
method), 93	aiogram.utils.formatting), 581
answer_voice() (aiogram.types.chat_join_request.ChatJ	
method), 96	aiogram.utils.formatting), 581
answer_voice() (aiogram.types.chat_member_updated.C	
method), 123	aiogram.utils.formatting), 581
	as_router() (aiogram.fsm.scene.Scene class method), 545
<pre>method), 196 answer_voice_pm() (aiogram.types.chat_join_request.Cl</pre>	
method), 97	AUD (aiogram.enums.currency.Currency attribute), 475
	AUDIO (aiogram.enums.content_type.ContentType at-
aiogram.methods.answer_callback_query),	tribute), 473
321	AUDIO (aiogram.enums.inline_query_result_type.InlineQueryResultType
answered (aiogram.utils.callback_answer.CallbackAnswer	
property), 579	AUDIO(aiogram.enums.input_media_type.InputMediaType
AnswerInlineQuery (class in	attribute), 480
aiogram.methods.answer_inline_query),	audio (aiogram.methods.send_audio.SendAudio at-
448	tribute), 378
AnswerPreCheckoutQuery (class in	audio(aiogram.types.external_reply_info.ExternalReplyInfo
<pre>aiogram.methods.answer_pre_checkout_query),</pre>	attribute), 129
456	audio (aiogram.types.message.Message attribute), 159
AnswerShippingQuery (class in	Audio (class in aiogram.types.audio), 19
aiogram.methods.answer_shipping_query), 457	audio_duration(aiogram.types.inline_query_result_audio.InlineQuery_attribute), 235
AnswerWebAppQuery (class in	audio_file_id(aiogram.types.inline_query_result_cached_audio.Inline
aiogram.methods.answer_web_app_query),	attribute), 237
450	<pre>audio_url(aiogram.types.inline_query_result_audio.InlineQueryResult_</pre>
ANY (aiogram.enums.content_type.ContentType at-	attribute), 234
tribute), 473	<pre>auth_date (aiogram.utils.web_app.WebAppInitData at-</pre>
<pre>api_url() (aiogram.client.telegram.TelegramAPIServer</pre>	tribute), 575
method), 13	<pre>author_signature (aiogram.types.message.Message</pre>
approve() (aiogram.types.chat_join_request.ChatJoinRed	
method), 61	$\verb author_signature   (a iogram. types. message\_origin\_channel. MessageOrigin\_channel. Mes$
ApproveChatJoinRequest (class in	attribute), 207
aiogram.methods.approve_chat_join_request),	author_signature(aiogram.types.message_origin_chat.MessageOrigin
322	attribute), 208
args (aiogram.filters.command.CommandObject at-	available_reactions (aiogram.types.chat.Chat
tribute), 500	attribute), 36
ARS (aiogram enums currency Currency attribute) 475	available reactions

(aiogram.types.chat_full_info.ChatFullInfo attribute), 58	BASKETBALL (aiogram.enums.dice_emoji.DiceEmoji attribute), 478
AZN (aiogram.enums.currency.Currency attribute), 475	BASKETBALL (aiogram.types.dice.DiceEmoji attribute), 127
В	BDT (aiogram.enums.currency.Currency attribute), 475
	BGN (aiogram.enums.currency.Currency attribute), 475
back() (aiogram.fsm.scene.SceneWizard method), 547	big_file_id (aiogram.types.chat_photo.ChatPhoto at-
background_custom_emoji_id	tribute), 126
(aiogram.types.chat.Chat attribute), 36	big_file_unique_id (aiogram.types.chat_photo.ChatPhoto
background_custom_emoji_id	attribute), 126
(aiogram.types.chat_full_info.ChatFullInfo	bio (aiogram.types.chat.Chat attribute), 36
attribute), 58	bio (aiogram.types.chat_full_info.ChatFullInfo at-
BackgroundFill (class in	
aiogram.types.background_fill), 20	tribute), 59
BackgroundFillFreeformGradient (class in	bio (aiogram.types.chat_join_request.ChatJoinRequest
aiogram.types.background_fill_freeform_gradien	t), attribute), 61
20	birthdate (aiogram.types.chat.Chat attribute), 36
BackgroundFillGradient (class in	birthdate (aiogram.types.chat_full_info.ChatFullInfo
aiogram.types.background_fill_gradient),	attribute), 58
21	Birthdate (class in aiogram.types.birthdate), 25
BackgroundFillSolid (class in	BLOCKQUOTE (aiogram.enums.message_entity_type.MessageEntityType
<pre>aiogram.types.background_fill_solid), 21</pre>	attribute), 481
BackgroundType (class in	BLUE (aiogram.enums.topic_icon_color.TopicIconColor
aiogram.types.background_type), 22	attribute), 484
BackgroundTypeChatTheme (class in	BND (aiogram.enums.currency.Currency attribute), 475
<pre>aiogram.types.background_type_chat_theme),</pre>	BOB (aiogram.enums.currency.Currency attribute), 475
22	BOLD (aiogram.enums.message_entity_type.MessageEntityType
BackgroundTypeFill (class in	attribute), 481
aiogram.types.background_type_fill), 23	Bold (class in aiogram.utils.formatting), 585
BackgroundTypePattern (class in	${\tt boost} (a iogram. types. chat\_boost\_up dated. ChatBoostUp dated$
aiogram.types.background_type_pattern),	attribute), 56
23	BOOST_ADDED (aiogram.enums.content_type.ContentType
BackgroundTypeWallpaper (class in	attribute), 474
aiogram.types.background_type_wallpaper),	boost_added (aiogram.types.message.Message at-
24	tribute), 161
BAM (aiogram.enums.currency.Currency attribute), 475	boost_count (aiogram.types.chat_boost_added.ChatBoostAdded
ban() (aiogram.types.chat.Chat method), 48	attribute), 53
ban_sender_chat() (aiogram.types.chat.Chat method),	boost_id (aiogram.types.chat_boost.ChatBoost at-
39	tribute), 52
BanChatMember (class in	boost_id(aiogram.types.chat_boost_removed.ChatBoostRemoved
aiogram.methods.ban_chat_member), 323	attribute), 53
·	boosts (aiogram.types.user_chat_boosts.UserChatBoosts
	attribute), 222
aiogram.methods.ban_chat_sender_chat), 325	bot_administrator_rights
DANK CTATEMENT ( -:	
attribute), 478	ement.Encr <b>/pie8FusspingEkeyboa</b> rd_button_request_chat.KeyboardButtonRe attribute), 149
base (aiogram.client.telegram.TelegramAPIServer	BOT_COMMAND (aiogram.enums.message_entity_type.MessageEntityType
attribute), 13	attribute), 481
BaseMiddleware (class in	$\verb bot_is_member  (a iogram. types. keyboard\_button\_request\_chat. KeyboardEnd and types. keyboardEnd $
aiogram.dispatcher.middlewares.base), 551	attribute), 149
BaseRequestHandler (class in	bot_username (aiogram.types.login_url.LoginUrl
aiogram.webhook.aiohttp_server), 513	attribute), 153
BaseSession (class in aiogram.client.session.base), 14	BotCommand (class in aiogram.types.bot_command), 25
BaseStorage (class in aiogram.fsm.storage.base), 530	BotCommand (class in aiogram.utils.formatting), 584
- · · · · · · · · · · · · · · · · · · ·	BotCommandScope (class in

aiogram.types.bot_command_scope), 25	attribute), 375
BotCommandScopeAllChatAdministrators (class in	
aiogram.types.bot_command_scope_all_chat_adr	ninistrator@jogram.methods.send_audio.SendAudio
26	attribute), 378
	business_connection_id
aiogram.types.bot_command_scope_all_group_ci	
26	attribute), 381
	business_connection_id
aiogram.types.bot_command_scope_all_private_o	
27	attribute), 382
•	business_connection_id
aiogram.types.bot_command_scope_chat), 27	(aiogram.methods.send_dice.SendDice at- tribute), 384
BotCommandScopeChatAdministrators (class in	
aiogram.types.bot_command_scope_chat_admini 28	strators), (aiogram.methods.send_document.SendDocument attribute), 387
<u>-</u>	business_connection_id
aiogram.types.bot_command_scope_chat_membe 28	r), (aiogram.methods.send_game.SendGame attribute), 453
BotCommandScopeDefault (class in	business_connection_id
<pre>aiogram.types.bot_command_scope_default), 29</pre>	(aiogram.methods.send_location.SendLocation attribute), 389
BotCommandScopeType (class in	business_connection_id
<pre>aiogram.enums.bot_command_scope_type), 471</pre>	(aiogram.methods.send_media_group.SendMediaGroup attribute), 391
BotDescription (class in	business_connection_id
aiogram.types.bot_description), 29	(aiogram.methods.send_message.SendMessage
BotName (class in aiogram.types.bot_name), 30 BotShortDescription (class in	attribute), 394 business_connection_id
BotShortDescription (class in aiogram.types.bot_short_description), 30	(aiogram.methods.send_photo.SendPhoto
bottom_color (aiogram.types.background_fill_gradient.Ba	
attribute), 21	business_connection_id
BOWLING (aiogram.enums.dice_emoji.DiceEmoji at-	(aiogram.methods.send_poll.SendPoll at-
tribute), 478	tribute), 399
	business_connection_id
BRL (aiogram.enums.currency.Currency attribute), 475	(aiogram.methods.send_sticker.SendSticker
BufferedInputFile (class in aiogram.types.input_file),	<pre>attribute), 310 business_connection_id</pre>
139, 488 build() (aiogram.fsm.storage.base.DefaultKeyBuilder	(aiogram.methods.send_venue.SendVenue
method), 530	attribute), 402
	business_connection_id
530	(aiogram.methods.send_video.SendVideo
build() (aiogram.utils.media_group.MediaGroupBuilder	attribute), 404
method), 590 BUSINESS_CONNECTION	business_connection_id
(aiogram.enums.update_type.UpdateType	(aiogram.methods.send_video_note.SendVideoNote attribute), 407
attribute), 484	business_connection_id
business_connection (aiogram.types.update.Update	(aiogram.methods.send_voice.SendVoice
attribute), 298	attribute), 410
business_connection_id	business_connection_id
(aiogram.methods.get_business_connection.GetBatribute), 348	usinessCon <b>(nėogiam</b> .types.business_messages_deleted.BusinessMessagesDel attribute), 32
	business_connection_id
(aiogram.methods.send_animation.SendAnimation	n (aiogram.types.message.Message attribute),

158	$\verb callback_game   (a iogram. types. in line\_key board\_button. In line Key boardButton. In line$
business_intro (aiogram.types.chat.Chat attribute),	attribute), 138
36 business_intro(aiogram.types.chat_full_info.ChatFullI	CALLBACK_QUERY (aiogram.enums.update_type.UpdateType Info attribute), 484
attribute), 58	callback_query (aiogram.types.update.Update at-
business_location (aiogram.types.chat.Chat at-	tribute), 299
tribute), 36	$\verb callback_query_id  (a iogram.methods.answer\_callback_query.AnswerCollege AnswerCollege AnswerCollege AnswerCollege AnswerCollege AnswerCollege AnswerDollege AnswerDol$
business_location(aiogram.types.chat_full_info.ChatF	
attribute), 58	callback_query_without_state
BUSINESS_MESSAGE (aiogram.enums.update_type.Update attribute), 484	547
business_message (aiogram.types.update.Update at- tribute), 298	CallbackAnswer (class in aiogram.utils.callback_answer), 579
business_opening_hours (aiogram.types.chat.Chat	
attribute), 36	CallbackAnswerMiddleware (class in
business_opening_hours	aiogram.utils.callback_answer), 579
(aiogram.types.chat_full_info.ChatFullInfo attribute), 58	CallbackData (class in aiogram.filters.callback_data), 506
BusinessConnection (class in aiogram.types.business_connection), 30	CallbackGame (class in aiogram.types.callback_game), 301
BusinessIntro (class in aiogram.types.business_intro),	CallbackQuery (class in
31	aiogram.types.callback_query), 33
	CallbackQueryHandler (class in
aiogram.types.business_location), 31	aiogram.handlers.callback_query), 558
	can_add_web_page_previews
aiogram.types.business_messages_deleted), 32 BusinessOpeningHours (class in	(aiogram.types.chat_member_restricted.ChatMemberRestricted attribute), 105
aiogram.types.business_opening_hours), 32	can_add_web_page_previews
BusinessOpeningHoursInterval (class in	(aiogram.types.chat_permissions.ChatPermissions attribute), 125
	),can_be_edited(aiogram.types.chat_member_administrator.ChatMember_
33	attribute), 100
button (aiogram.methods.answer_inline_query.AnswerIntattribute), 449	lina@u.change_info(aiogram.methods.promote_chat_member.PromoteChatattribute), 368
button_text(aiogram.types.web_app_data.WebAppData attribute), 228	a can_change_info(aiogram.types.chat_administrator_rights.ChatAdminis attribute), 51
buttons (aiogram.utils.keyboard.InlineKeyboardBuilder property), 564	can_change_info(aiogram.types.chat_member_administrator.ChatMembattribute), 101
	can_change_info(aiogram.types.chat_member_restricted.ChatMemberReattribute), 105
BYN (aiogram.enums.currency.Currency attribute), 475	can_change_info(aiogram.types.chat_permissions.ChatPermissions attribute), 125
C	can_connect_to_business (aiogram.types.user.User
cache_time(aiogram.methods.answer_callback_query.Anattribute), 321	can_delete_messages
cache_time (aiogram.methods.answer_inline_query.Answattribute), 449	werInlineQueriogram.methods.promote_chat_member.PromoteChatMember attribute), 368
cache_time (aiogram.utils.callback_answer.CallbackAnswer.Callback)	<sub>W.E.</sub> an_delete_messages
property), 580	(aiogram.types.chat_administrator_rights.ChatAdministratorRigh
CAD (aiogram.enums.currency.Currency attribute), 475	attribute), 51
callback_data(aiogram.handlers.callback_query.Callback_property), 558	(aiogram.types.chat_member_administrator.ChatMemberAdminis
callback_data(aiogram.types.inline_keyboard_button.Inattribute), 138	nlineKeybod#tl#bl#6h <sup>100</sup> can_delete_stories(aiogram.methods.promote_chat_member.Promote0

```
attribute), 369
                                                                                                                                                                                                                                                                                                                                      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
can_delete_stories (aiogram.types.chat_administrator_rights.Chattathinia)trator_rights
                                                                                                                                                                                                                                                                                                                                       can_pin_messages (aiogram.types.chat_administrator_rights.ChatAdmin
                                                     attribute), 51
can_delete_stories(aiogram.types.chat_member_administrator.ChattMbanel)e5Administrator
                                                      attribute), 101
                                                                                                                                                                                                                                                                                                                                       can_pin_messages (aiogram.types.chat_member_administrator.ChatMem
can_edit_messages(aiogram.methods.promote_chat_member.Promotte@hateMember
                                                     attribute), 369
                                                                                                                                                                                                                                                                                                                                       can_pin_messages (aiogram.types.chat_member_restricted.ChatMember_
 can_edit_messages(aiogram.types.chat_administrator_rights.ChatAtdriibinuixat);atOfRights
                                                      attribute), 51
                                                                                                                                                                                                                                                                                                                                      \verb|can_pin_messages| (a iogram. types. chat\_permissions. Chat Permissions|) \\
can_edit_messages (aiogram.types.chat_member_administrator.ChattYlbmule\)rAdvinistrator
                                                      attribute), 101
                                                                                                                                                                                                                                                                                                                                       {\tt can\_post\_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.PromoteC
 can_edit_stories(aiogram.methods.promote_chat_member.Promote@HattMgmb0r
                                                     attribute), 369
                                                                                                                                                                                                                                                                                                                                      can_post_messages (aiogram.types.chat_administrator_rights.ChatAdmi
 can_edit_stories (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.ChatAdministr
                                                                                                                                                                                                                                                                                                                                       \verb|can_post_messages|| (a iogram. types. chat\_member\_administrator. Chat Messages)|| (a iogram. types. chat\_messages)|| (a iogram.
                                                      attribute), 51
 can_edit_stories (aiogram.types.chat_member_administrator.ChatMribber)Administrator
                                                      attribute), 101
                                                                                                                                                                                                                                                                                                                                      can_post_stories(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_
can_invite_users(aiogram.methods.promote_chat_member.Promote@HutMgmb0r
                                                                                                                                                                                                                                                                                                                                      can_post_stories(aiogram.types.chat_administrator_rights.ChatAdmin
                                                      attribute), 369
can_invite_users (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.ChatAdministr
                                                      attribute), 51
                                                                                                                                                                                                                                                                                                                                       can_post_stories(aiogram.types.chat_member_administrator.ChatMem
can_invite_users (aiogram.types.chat_member_administrator.ChatMeibloter)Administrator
                                                       attribute), 101
                                                                                                                                                                                                                                                                                                                                       can_promote_members
can_invite_users (aiogram.types.chat_member_restricted.ChatMetalbegRestmathbds.promote_chat_member.PromoteChatMember
                                                      attribute), 105
                                                                                                                                                                                                                                                                                                                                                                                              attribute), 368
can_invite_users (aiogram.types.chat_permissions.Chat@annipromoste_members
                                                       attribute), 125
                                                                                                                                                                                                                                                                                                                                                                                              (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), 51
can_join_groups (aiogram.types.user.User attribute),
                                                                                                                                                                                                                                                                                                                                       can_promote_members
{\tt can\_manage\_chat} \ (a iogram.methods.promote\_chat\_member.Promote \ \textit{ColorgitMentlype} s. chat\_member\_administrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatMemberAdministrator.ChatM
                                                     attribute), 368
                                                                                                                                                                                                                                                                                                                                                                                               attribute), 101
can_manage_chat (aiogram.types.chat_administrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministrator_rightsaf\meadministra
                                                     attribute), 50
                                                                                                                                                                                                                                                                                                                                                                                               (aiogram.types.user.User attribute), 221
{\tt can\_manage\_chat} \ (a iogram. types. chat\_member\_administr {\tt aam\_GNeptMyd} (ulbegr {\tt Audmitypiets: dottosiness\_connection}. Business {\tt Connection}
                                                      attribute), 100
                                                                                                                                                                                                                                                                                                                                                                                               attribute), 31
can_manage_topics (aiogram.methods.promote_chat_member_PrestroieChamberbelns
                                                     attribute), 369
                                                                                                                                                                                                                                                                                                                                                                                              (aiogram.methods.promote_chat_member.PromoteChatMember
can_manage_topics(aiogram.types.chat_administrator_rights.ChatAtdmibiuitxt);a168Rights
                                                      attribute), 51
                                                                                                                                                                                                                                                                                                                                       can_restrict_members
can_manage_topics (aiogram.types.chat_member_administrator.Ch(atMgnalmentAplensichlustr_audministrator_rights.ChatAdministratorRights.ChatAdministrator.Ch(atMgnalmentAplensichlustr_audministrator_rights.ChatAdministratorRights.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.ChatAdministrator.C
                                                                                                                                                                                                                                                                                                                                                                                              attribute), 51
                                                     attribute), 101
can_manage_topics (aiogram.types.chat_member_restrictedrC/metMenibetr_Restricted
                                                      attribute), 105
                                                                                                                                                                                                                                                                                                                                                                                              (aiogram.types.chat_member_administrator.ChatMemberAdminis
 can_manage_topics (aiogram.types.chat_permissions.ChatPermissiattsribute), 101
                                                                                                                                                                                                                                                                                                                                       {\tt can\_send\_after}\ (aiogram.utils.web\_app.WebAppInitData
                                                     attribute), 125
 can_manage_video_chats
                                                                                                                                                                                                                                                                                                                                                                                               attribute), 575
                                                     (aiogram.methods.promote_chat_member.Promote@hmats4emdbaadios (aiogram.types.chat_member_restricted.ChatMemberR
                                                     attribute), 368
                                                                                                                                                                                                                                                                                                                                                                                              attribute), 104
can_manage_video_chats
                                                                                                                                                                                                                                                                                                                                       can_send_audios (aiogram.types.chat_permissions.ChatPermissions
                                                       (aiogram.types.chat_administrator_rights.ChatAdministratoaRightse), 124
                                                     attribute), 51
                                                                                                                                                                                                                                                                                                                                       can_send_documents(aiogram.types.chat_member_restricted.ChatMemb
 can_manage_video_chats
                                                                                                                                                                                                                                                                                                                                                                                              attribute), 104
                                                      (aiogram.types.chat_member_administrator.ChatManbæeAdndoistmeonts (aiogram.types.chat_permissions.ChatPermissions
                                                                                                                                                                                                                                                                                                                                                                                              attribute), 124
                                                     attribute), 100
```

can_send_messages(aiogram.types.chat_member_restric	ted ChatMatthibriRensthikfted
attribute), 104	${\tt caption} (aiogram. types. in line\_query\_result\_audio. In lineQueryResultAudio. In lineQuer$
can_send_messages (aiogram.types.chat_permissions.Chat_	
<pre>attribute), 124 can_send_other_messages</pre>	<pre>caption(aiogram.types.inline_query_result_cached_audio.InlineQueryR</pre>
	,,,
attribute), 105	nbapRasmi(aidgram.types.inline_query_result_cached_document.InlineQue attribute), 239
can_send_other_messages	<pre>caption(aiogram.types.inline_query_result_cached_gif.InlineQueryResu</pre>
(aiogram.types.chat_permissions.ChatPermission	
	<pre>caption(aiogram.types.inline_query_result_cached_mpeg4_gif.InlineQu</pre>
can_send_photos(aiogram.types.chat_member_restricted	
- · · · · · · · · · · · · · · · · · · ·	caption (aiogram.types.inline_query_result_cached_photo.InlineQueryR
can_send_photos (aiogram.types.chat_permissions.ChatF	
	caption (aiogram.types.inline_query_result_cached_video.InlineQueryRe
can_send_polls (aiogram.types.chat_member_restricted.	
	caption (aiogram.types.inline_query_result_cached_voice.InlineQueryRe
can_send_polls (aiogram.types.chat_permissions.ChatPe	
	caption (aiogram.types.inline_query_result_document.InlineQueryResult_
can_send_video_notes	attribute), 255
	nbapRismi(uidgram.types.inline_query_result_gif.InlineQueryResultGif
attribute), 104	attribute), 258
	caption (aiogram.types.inline_query_result_mpeg4_gif.InlineQueryResu
(aiogram.types.chat_permissions.ChatPermission	
	caption (aiogram.types.inline_query_result_photo.InlineQueryResultPho
can_send_videos (aiogram.types.chat_member_restricted	
attribute), 104 can_send_videos (aiogram.types.chat_permissions.ChatF	caption (aiogram.types.inline_query_result_video.InlineQueryResultVideo.apticipute)
	caption (aiogram.types.inline_query_result_voice.InlineQueryResultVoice.
can_send_voice_notes	attribute), 269
	nbapRasmi(nidgram.types.input_media_animation.InputMediaAnimation
attribute), 105	attribute), 141
	caption (aiogram.types.input_media_audio.InputMediaAudio
(aiogram.types.chat_permissions.ChatPermission	
	caption (aiogram.types.input_media_document.InputMediaDocument
can_set_sticker_set (aiogram.types.chat.Chat	attribute), 143
	caption (aiogram.types.input_media_photo.InputMediaPhoto
can_set_sticker_set	attribute), 144
(aiogram.types.chat_full_info.ChatFullInfo attribute), 60	caption (aiogram.types.input_media_video.InputMediaVideo attribute), 145
<pre>caption (aiogram.methods.copy_message.CopyMessage</pre>	caption (aiogram.types.message.Message attribute), 160
	caption_entities(aiogram.methods.copy_message.CopyMessage
caption(aiogram.methods.edit_message_caption.EditMes.	
	caption_entities(aiogram.methods.edit_message_caption.EditMessag
caption(aiogram.methods.send_animation.SendAnimation	- 0 = 1
attribute), 376	caption_entities(aiogram.methods.send_animation.SendAnimation
caption (aiogram.methods.send_audio.SendAudio at-	attribute), 376
	caption_entities(aiogram.methods.send_audio.SendAudio
caption (aiogram.methods.send_document.SendDocument	
	caption_entities(aiogram.methods.send_document.SendDocument
caption (aiogram.methods.send_photo.SendPhoto at-	attribute), 387
	caption_entities(aiogram.methods.send_photo.SendPhoto
caption (aiogram.methods.send_video.SendVideo	attribute), 396
	caption_entities(aiogram.methods.send_video.SendVideo
caption (aiogram.methods.send_voice.SendVoice	attribute), 405
- · · · · · · · · · · · · · · · · · · ·	**

chat\_id(aiogram.methods.approve chat join request.ApproveChatJoinRe

```
caption_entities(aiogram.methods.send_voice.SendVoichannel_chat_created
                                                                                                                                                                                                                                                     attribute),
                      attribute), 410
                                                                                                                                                                (aiogram.types.message.Message
caption_entities (aiogram.types.inline_query_result_audio.Inline QueryResultAudio
                                                                                                                                        CHANNEL_POST (aiogram.enums.update_type.UpdateType
                      attribute), 235
caption_entities(aiogram.types.inline_query_result_cached_audiatthibimaeQu484ResultCachedAudio
                      attribute), 237
                                                                                                                                        channel_post (aiogram.types.update.Update attribute),
caption_entities(aiogram.types.inline_query_result_cached_document.InlineQueryResultCachedDocument
                                                                                                                                         CHAT (aiogram.enums.bot_command_scope_type.BotCommandScopeType
                      attribute), 239
caption_entities (aiogram.types.inline_query_result_cached_gif.lattrieQue); AResultCachedGif
                                                                                                                                        CHAT (aiogram.enums.message_origin_type.MessageOriginType
                      attribute), 241
caption_entities(aiogram.types.inline_query_result_cached_mpegtfribilith)lih8QueryResultCachedMpeg4Gif
                                                                                                                                        chat (aiogram.types.business_messages_deleted.BusinessMessagesDeleted
                      attribute), 243
caption\_entities (a log ram. types. in line\_query\_result\_cached\_photatthibinteQueryResultCachedPhotological and the properties of the pr
                                                                                                                                         chat (aiogram.types.chat_boost_removed.ChatBoostRemoved
                      attribute), 245
caption_entities(aiogram.types.inline_query_result_cached_videontributeQuteryResultCachedVideo
                      attribute), 249
                                                                                                                                         \verb|chat|| (a iogram.types.chat\_boost\_updated.ChatBoostUpdated|
caption_entities(aiogram.types.inline_query_result_cached_voicattribineQuivbyResultCachedVoice
                                                                                                                                        chat (aiogram.types.chat_join_request.ChatJoinRequest
                      attribute), 251
caption_entities(aiogram.types.inline_query_result_document.Indition)yResultDocument
                                                                                                                                        chat (aiogram.types.chat_member_updated.ChatMemberUpdated
                      attribute), 255
caption_entities (aiogram.types.inline_query_result_gif.InlineQuartyResult)Giff5
                      attribute), 258
                                                                                                                                        \verb|chat| (a iogram.types.external\_reply\_info.ExternalReplyInfo|\\
caption_entities(aiogram.types.inline_query_result_mpeg4_gif.lntinitQue);yResultMpeg4Gif
                      attribute), 262
                                                                                                                                        chat (aiogram.types.giveaway_winners.GiveawayWinners
caption_entities (aiogram.types.inline_query_result_photo.Inline Query_utile) to the caption_entities (aiogram.types.inline_query_result_photo.Inline Query_result_photo.Inline Query_result_photo.Inlin
                      attribute), 264
                                                                                                                                        \verb|chat| (a iogram.types.in accessible \_message.In accessible Message|
caption_entities (aiogram.types.inline_query_result_video.Inline QueibyRe);ultVideo
                                                                                                                                         chat (aiogram.types.message.Message attribute), 158
                      attribute), 268
caption_entities (aiogram.types.inline_query_result_voithealtulinie@numxtRpsslttVexissage_origin_channel.MessageOriginChannel
                      attribute), 270
                                                                                                                                                               attribute), 207
caption_entities(aiogram.types.input_media_animationc\text{Mapu(tMkeghiarhutiypasimessage_reaction_count_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCount_updated.MessageReactionCoun
                      attribute), 141
                                                                                                                                                                attribute), 209
caption_entities(aiogram.types.input_media_audio.Inpublifie\fainityqudiin.types.message_reaction_updated.MessageReactionUpdated
                                                                                                                                                               attribute), 210
                      attribute), 142
caption_entities (aiogram.types.input_media_documenthquulMediadhotypex.story.Story attribute), 219
                                                                                                                                        chat (aiogram.utils.web_app.WebAppInitData attribute),
                      attribute), 143
caption_entities(aiogram.types.input_media_photo.InputMediaPhoto
                      attribute), 144
                                                                                                                                         Chat (class in aiogram.types.chat), 35
caption_entities(aiogram.types.input_media_video.Inp@HMTedADMTUNESTRATORS
                                                                                                                                                               (aiogram.enums.bot_command_scope_type.BotCommandScopeTy
                      attribute), 145
                                                    (aiogram.types.message.Message
caption_entities
                                                                                                                                                               attribute), 471
                                                                                                                                         CHAT_BACKGROUND_SET
                      attribute), 160
CASHTAG (aiogram.enums.message_entity_type.MessageEntityType
                                                                                                                                                              (aiogram.enums.content_type.ContentType
                      attribute), 481
                                                                                                                                                                attribute), 474
CashTag (class in aiogram.utils.formatting), 584
                                                                                                                                         chat_background_set
certificate(aiogram.methods.set_webhook.SetWebhook
                                                                                                                                                                (aiogram.types.message.Message
                                                                                                                                                                                                                                                     attribute),
                                                                                                                                                                162
                      attribute), 468
CHANNEL (aiogram.enums.chat_type.ChatType attribute), CHAT_BOOST (aiogram.enums.update_type.UpdateType
                                                                                                                                                                attribute), 485
CHANNEL (aiogram.enums.message_origin_type.MessageOrighinallyphroost (aiogram.types.update.Update attribute),
                      attribute), 482
CHANNEL_CHAT_CREATED
                                                                                                                                         chat_has_username(aiogram.types.keyboard_button_request_chat.Keybo
                      (aiogram.enums.content_type.ContentType
                                                                                                                                                                attribute), 149
```

Index 647

attribute), 474

attribute), 322 attribute), 350 chat\_id (aiogram.methods.ban\_chat\_member.BanChatMembert\_id (aiogram.methods.get\_chat\_member.GetChatMember attribute), 351 attribute), 323 chat\_id (aiogram.methods.ban\_chat\_sender\_chat.BanChatSentleid((atogram.methods.get\_chat\_member\_count.GetChatMemberCount attribute), 325 attribute), 352 chat\_id(aiogram.methods.close forum topic.CloseForumEbait\_id(aiogram.methods.get chat menu button.GetChatMenuButton attribute), 353 attribute), 327 chat\_id (aiogram.methods.close\_general\_forum\_topic.Close\Genetate\distribute\text{Gene}\distribute\text{Disorgnathopnie} thods.get\_game\_high\_scores.GetGameHighScores attribute), 328 attribute), 452 chat\_id (aiogram.methods.copy\_message.CopyMessage chat\_id (aiogram.methods.get\_user\_chat\_boosts.GetUserChatBoosts attribute), 329 attribute), 361 chat\_id (aiogram.methods.copy\_messages.CopyMessages chat\_id (aiogram.methods.hide\_general\_forum\_topic.HideGeneralForum) attribute), 331 attribute), 363 chat\_id (aiogram.methods.create\_chat\_invite\_link.Create@hatfnyiteLink (aiogram.methods.leave\_chat.LeaveChat attribute), 364 attribute), 333 chat\_id(aiogram.methods.create\_forum\_topic.CreateForumHapiod(aiogram.methods.pin\_chat\_message.PinChatMessage attribute), 334 attribute), 366 chat\_id (aiogram.methods.decline\_chat\_join\_request.DeclaracthatddintRequestmethods.promote\_chat\_member.PromoteChatMember attribute), 335 attribute), 368 chat\_id(aiogram.methods.delete chat photo.DeleteChatPhhat\_id(aiogram.methods.reopen forum topic.ReopenForumTopic attribute), 336 attribute), 370 chat\_id (aiogram.methods.delete\_chat\_sticker\_set.DeleteChhasticket,Sieogram.methods.reopen\_general\_forum\_topic.ReopenGeneralFo attribute), 371 attribute), 337 chat\_id(aiogram.methods.delete forum topic.DeleteForumWayicid(aiogram.methods.restrict chat member.RestrictChatMember attribute), 338 attribute), 372 chat\_id (aiogram.methods.delete\_message.DeleteMessage chat\_id (aiogram.methods.revoke\_chat\_invite\_link.RevokeChatInviteLink attribute), 435 attribute), 373 chat\_id (aiogram.methods.delete\_messages.DeleteMessageshat\_id (aiogram.methods.send\_animation.SendAnimation attribute), 436 attribute), 375 chat\_id(aiogram.methods.edit\_chat\_invite\_link.EditChatInviaeLink (aiogram.methods.send\_audio.SendAudio atattribute), 341 *tribute*), 378 chat\_id(aiogram.methods.edit\_forum\_topic.EditForumTopihat\_id(aiogram.methods.send\_chat\_action.SendChatAction attribute), 342 attribute), 380 chat\_id (aiogram.methods.edit\_general\_forum\_topic.EditCahartalidorumiTopicam.methods.send\_contact.SendContact attribute), 343 attribute), 382 chat\_id(aiogram.methods.edit message caption.EditMess@ac@idion(aiogram.methods.send dice.SendDice attribute), 437 tribute), 384 chat\_id (aiogram.methods.edit\_message\_live\_location.EditMaxsageLiveetmenthods.send\_document.SendDocument attribute), 439 attribute), 386 chat\_id(aiogram.methods.edit\_message\_media.EditMessaghMedia (aiogram.methods.send\_game.SendGame attribute), 441 attribute), 453 chat\_id (aiogram.methods.edit\_message\_reply\_markup.EditMatssingaRipplyMathathods.send\_invoice.SendInvoice atattribute), 442 tribute), 461 chat\_id(aiogram.methods.edit\_message\_text.EditMessage@that\_id (aiogram.methods.send\_location.SendLocation attribute), 444 attribute), 389 chat\_id (aiogram.methods.export\_chat\_invite\_link.ExportChatlniiddhiiolgram.methods.send\_media\_group.SendMediaGroup attribute), 344 attribute), 391 chat\_id(aiogram.methods.forward\_message.ForwardMessageat\_id (aiogram.methods.send\_message.SendMessage attribute), 345 attribute), 393 chat\_id(aiogram.methods.forward\_messages.ForwardMesshges\_id (aiogram.methods.send\_photo.SendPhoto atattribute), 347 tribute), 396 chat\_id (aiogram.methods.get\_chat.GetChat attribute), chat\_id (aiogram.methods.send poll.SendPoll 349 tribute), 398

648 Index

chat\_id(aiogram.methods.get chat administrators.GetChathathnindstynatogsam.methods.send sticker.SendSticker at-

tribute), 310	attribute), 217			
chat_id (aiogram.methods.send_venue.SendVenue at-		.callback_query.Ca	ıllbackQuery	
tribute), 401	attribute), 33			
chat_id (aiogram.methods.send_video.SendVideo		web_app.WebAppIr	ıitData	
attribute), 404	attribute), 574			
chat_id(aiogram.methods.send_video_note.SendVideoN	<i>Note</i> hat_is_channel( <i>aiogram.typ</i>	oes.keyboard_buttoi	n_request_chat.I	Keyboai
attribute), 407	attribute), 148			
${\tt chat\_id}$ $(aiogram.methods.send\_voice.SendVoice)$	chat_is_created(aiogram.typ	oes.keyboard_buttoi	n_request_chat.	Keyboai
attribute), 409	attribute), 149			
${\tt chat\_id}\ (aiogram.methods.set\_chat\_administrator\_custor)$	om <u>chidae. Sie tCl FoodAuduv (ioniico greatron: G</u> ypsetx	o <b>keTjhb</b> ard_button_1	request_chat.Ke <sub>?</sub>	yboardE
attribute), 412	attribute), 148			
${\tt chat\_id}(aiogram.methods.set\_chat\_description.SetChat$		enums.update_type	.UpdateType	
attribute), 413	attribute), 485			
chat_id(aiogram.methods.set_chat_menu_button.SetCh	<i>at<b>Mhaπ</b>B</i> ព្ <b>ច</b> ច់ <b>n</b> _request ( <i>aiogram</i>	.types.update.Upda	te at-	
attribute), 414	tribute), 299			
${\tt chat\_id}\ (aiogram.methods.set\_chat\_permissions.SetChat\_id)$	tPGHATSMENBER (aiogram.enums.l	pot_command_scop	e_type.BotComr	nandSco
attribute), 415	attribute), 471			
${\tt chat\_id}(aiogram.methods.set\_chat\_photo.SetChatPhoto)$		.update_type.Updat	еТуре	
attribute), 416	attribute), 485			
chat_id(aiogram.methods.set_chat_sticker_set.SetChatS	Stiddatemember (aiogram.types.u	ıpdate.Update attri	bute),	
attribute), 417	299			
chat_id (aiogram.methods.set_chat_title.SetChatTitle		content_type.Conter	ıtType	
attribute), 418	attribute), 474			
${\tt chat\_id}(aiogram.methods.set\_game\_score.SetGameScore)$		s.message.Message	at-	
attribute), 455	tribute), 161			
${\tt chat\_id}\ (aiogram.methods.set\_message\_reaction.SetMessage\_reaction)$		ıe_query.InlineQue	ry at-	
attribute), 419	tribute), 230			
chat_id(aiogram.methods.stop_message_live_location.S		_app.WebAppInitDa	ta at-	
attribute), 445	tribute), 574			
chat_id (aiogram.methods.stop_poll.StopPoll attribute),	_			
447	ChatActionMiddleware	(class	in	
chat_id(aiogram.methods.unban_chat_member.UnbanC				
attribute), 426	ChatActionSender (class in ai	ogram.utils.chat_ac	ction),	
chat_id(aiogram.methods.unban_chat_sender_chat.Unl				
attribute), 428	ChatAdministratorRights	(class	in	
chat_id(aiogram.methods.unhide_general_forum_topic.	~	lministrator_rights)	,	
attribute), 429	50			
chat_id(aiogram.methods.unpin_all_chat_messages.Un		(class	ın	
attribute), 430	aiogram.types.chat_ba			
chat_id(aiogram.methods.unpin_all_forum_topic_mess		•		
attribute), 431	ChatBoostAdded	(class	in	
chat_id(aiogram.methods.unpin_all_general_forum_top	0 01 11			
attribute), 432	ChatBoostRemoved	(class	in	
Chat_id(aiogram.methods.unpin_chat_message.UnpinC				
attribute), 433	ChatBoostSource	(class	in	
<pre>chat_id(aiogram.types.bot_command_scope_chat.BotC</pre>	ChatBoostSourceGiftCode	(class	in	
chat_id(aiogram.types.bot_command_scope_chat_adm		`		
attribute), 28	111131141018.12 <b>01009141114119418121141</b> 54	iBM <u>in</u> dicults to ea <u>r</u> gript_co	ue),	
chat_id(aiogram.types.bot_command_scope_chat_mem	- ·	er (class	in	
attribute), 29	aiogram.types.chat_bo			
chat_id (aiogram.types.chat_shared.ChatShared at-	~ **	.osi_somec_giveaw	<i>wy 1</i> ,	
tribute), 126	ChatBoostSourcePremium	(class	in	
chat_id(aiogram.types.reply parameters.ReplyParamet		`		

55		CHOSEN_INLINE_RESULT
ChatBoostSourceType (class	in	(aiogram.enums.update_type.UpdateType
aiogram.enums.chat_boost_source_type),		attribute), 484
472		<pre>chosen_inline_result (aiogram.types.update.Update</pre>
ChatBoostUpdated (class	in	attribute), 299
<pre>aiogram.types.chat_boost_updated), 56</pre>		ChosenInlineResult (class in
ChatFullInfo (class in aiogram.types.chat_full_inf	fo),	aiogram.types.chosen_inline_result), 229
56		city (aiogram.types.shipping_address.ShippingAddress
ChatInviteLink (class	in	attribute), 295
aiogram.types.chat_invite_link), 60	·	clear_data() (aiogram.fsm.scene.SceneWizard
ChatJoinRequest (class	in	method), 547
aiogram.types.chat_join_request), 61	)	ClientDecodeError, 555
ChatLocation (class in aiogram.types.chat_locatio	m),	
Cha+Mombor (class in giogram types chat member) 0	0	close() (aiogram.client.session.base.BaseSession method), 14
ChatMember (class in aiogram.types.chat_member), 9 ChatMemberAdministrator (class		close() (aiogram.fsm.scene.ScenesManager method),
aiogram.types.chat_member_administrator),		546
99	,	close() (aiogram.fsm.storage.base.BaseStorage
ChatMemberBanned (class	in	method), 531
aiogram.types.chat_member_banned), 102	ııı	close() (aiogram.webhook.aiohttp_server.SimpleRequestHandler
ChatMemberLeft (class	in	method), 514
aiogram.types.chat_member_left), 102	uu	close_date (aiogram.methods.send_poll.SendPoll at-
ChatMemberMember (class	in	tribute), 399
aiogram.types.chat_member_member), 103		close_date (aiogram.types.poll.Poll attribute), 212
ChatMemberOwner (class	in	CloseForumTopic (class in
aiogram.types.chat_member_owner), 103		aiogram.methods.close_forum_topic), 327
ChatMemberRestricted (class	in	CloseGeneralForumTopic (class in
aiogram.types.chat_member_restricted),		aiogram.methods.close_general_forum_topic),
104		328
ChatMemberStatus (class	in	closing_minute(aiogram.types.business_opening_hours_interval.Busine
aiogram.enums.chat_member_status), 472		attribute), 33
ChatMemberUpdated (class	in	CLP (aiogram.enums.currency.Currency attribute), 475
aiogram.types.chat_member_updated), 105		CNY (aiogram.enums.currency.Currency attribute), 475
ChatMemberUpdatedFilter (class	in	CODE (aiogram.enums.message_entity_type.MessageEntityType
aiogram.filters.chat_member_updated), 501		attribute), 481
ChatPermissions (class	in	Code (class in aiogram.utils.formatting), 585
aiogram.types.chat_permissions), 124		color (aiogram.types.background_fill_solid.BackgroundFillSolid
ChatPhoto (class in aiogram.types.chat_photo), 125		attribute), 22
chats (aiogram.types.giveaway.Giveaway attribute), 1		colors (aiogram.types.background_fill_freeform_gradient.BackgroundFill_
ChatShared (class in aiogram.types.chat_shared), 120	6	attribute), 21
ChatType (class in aiogram.enums.chat_type), 473		command (aiogram.filters.command.CommandObject at-
check_flags() (in module aiogram.dispatcher.flag	ζs),	tribute), 500
556	~	command (aiogram.types.bot_command.BotCommand at-
check_response() (aiogram.client.session.base.Base	eSes.	
method), 14	,	Command (class in aiogram.filters.command), 499
check_webapp_signature() (in mode	ше	CommandObject (class in aiogram.filters.command), 500
aiogram.utils.web_app), 573	_	COMMANDS (aiogram.enums.menu_button_type.MenuButtonType
CHTN (aiogram.enums.currency.Currency attribute), 475		attribute), 481
	wnP	oʻoʻoʻoʻmmands (aiogram.methods.set_my_commands.SetMyCommands attribute), 421
attribute), 480	ction	attribute), 421 n CONNECTED_WEBSITE (aiogram.enums.content_type.ContentType
attribute), 472	cuor	attribute), 474
	ctio	n&enthercted_website (aiogram.types.message.Message
class method), 570		attribute), 161

CONTACT (alogram.emus.content_type.ContentType arribute), 473 CONTACT (alogram.emus.minine_query_result_type.InlineQueryResultBypeute, 473 CONTACT (alogram.emus.minine_query_result_type.InlineQueryResultBypeute, 473 Contact (alogram.types.external_reply_info.ExternalReply.hfo attribute), 130 Contact (class in alogram.types.contact), 127 Content_type (alogram.types.message.Message arribute), 160 Contact (class in alogram.types.message.Message property), 163 ContentType (alogram.types.message.Message property), 163 Copy () (alogram.emus.currency.Currency attribute), 475 Copy () (alogram.types.message.Message method), 564 Copy () (alogram.types.message.Message method), 565 Copy_to() (alogra	ConstI18nMiddleware (class in aiogram.utils.i18n.middleware), 567	<pre>creates_join_request     (aiogram.types.chat_invite_link.ChatInviteLink)</pre>
CONTACT (alogram.emus inline_query_result_type.hullneQuery_ResultImpoute), 472	CONTACT (aiogram.enums.content_type.ContentType at-	attribute), 60
creator (alogram.types.chat_invite_link.ChathwiteLink contact (alogram.types.message.Message attribute), 160 contact (class in alogram.types.message.Message attribute), 160 contact (class in alogram.types.message.Message property), 163 content.type (alogram.types.message.Message property), 163 content.type (alogram.types.that.chat.thine.Keyboard.Builder method), 564 copy () (alogram.types.that.seyboard.Reply.KeyboardBuilder method), 565 copy_to() (alogram.types.sage.Message method), 198 copyMessage (class in alogram.methods.copy_message), 331 copyMessage (class in alogram.methods.copy_message), 331 corpyto() (alogram.types.that.chat.thine.types.go) alogram.methods.copy_message), 331 corpyto() (alogram.types.that.theat.thine.types.go) corpy_to() (alogram.types.that.theat.thine.types.go) alogram.methods.copy_message), 331 corpyto() (alogram.types.that.theat.thine.types.go) corpy_to() (alogram.types.that.thine.types.go) corpy_to() (alogram.types.that.thine.thine.types.go) corpy_to() (alogram.types.that.thine.thine.types.go) corpy_to() (alogram.types.that.thine.thine.thine.thine.types.go) corpy_to() (alogram.types.that.thine.thine.thine.thine.thine.thine.thine.thine.thine.thine.thine.thine.thine.thine.thin		
contact (alogram.types.exerend _reply_injo.ExternalReplytho artiribure), 130 contact (alogram.types.message.Message attribute), 163 contact (class in alogram.types.message.Message property), 163 contentType (alogram.types.message.Message property), 163 contentType (class in alogram.enums.cortent_type), 473 copy() (alogram.types.message.Message property), 163 copy() (alogram.types.message.Message property), 163 copy() (alogram.types.message.Message property), 163 copy() (alogram.types.message.Message property), 163 copy() (alogram.types.message.Message method), 475 copy() (alogram.types.message.Message method), 565 copy_to() (alogram.types.message.Message method), 198 copy() (alogram.types.message.Message.Message.Message.Message.Message.Message.Message.Message.M		
createntials (alogram.types.passport_data.PassportData attribute), 130 Contact (class in alogram.types.message.Message properry), 163 Content_type (alogram.types.message.Message properry), 163 Content_type (class in alogram.enums.content_type), 473 COP (alogram.types.message.Message properry), 163 COP (alogram.types.message.Message method), 564 Copy() (alogram.types.message.Message method), 564 Copy() (alogram.types.message.Message method), 565 Copy_to() (alogram.types.message.Message method), 198 CopyMessage (class in alogram.methods.copy_message), 199 Copy	· · · · · · · · · · · · · · · · · · ·	
contact (alogram.types.message.Message attribute), 160 content type (alogram.types.message.Message properry), 163 content type (class in alogram.enums.content_type), 473 content type (class in alogram.enums.content_type), 473 copy (alogram.enums.currency.Currency attribute), 475 copy (alogram.enum.content_type), 473 copy (alogram.enum.content_type), 475 copy (alogram.enum.content.content_type), 475 copy (alogram.enum.content		
Content_type (alogram.types.contact). 127 content_type (class in alogram.enums.content_type). 473 COP (alogram.enums.currency Currency attribute). 475 copy) (alogram.attlis.keyboard.lnline.keyboardBuilder method). 565 copy_() (alogram.types.message.Message method). 198 copy) (alogram.types.message.Message.Message.method). 209 copy) (alogram.types.message.Mes		
contentType (alogram.types.message.Message property), 163 ContentType (class in alogram.emums.content_type), 473 COP (alogram.emums.currency.Currency attribute), 475 copy() (alogram.utils.keyboard.InlineKeyboardBuilder method), 564 copy() (alogram.utils.keyboard.ReplyKeyboardBuilder method), 565 copy_to() (alogram.types.message.Message method), 198 CopyMessage (class in alogram.methods.copy_message), 198 CopyMessage (class in alogram.methods.copy_message), 198 CopyMessage (class in alogram.methods.copy_message), 229 CopyMessage (class in alogram.methods.copy_message), 331 correct_option_id (alogram.types.goll.Poll attribute), 399 correct_option_id (alogram.types.poll.Poll attribute), 399 correct_option_id (alogram.types.poll.Poll attribute), 395 country_code (alogram.types.giveaway.Giveaway attribute), 135 CRC (alogram.methods.create_chat_invite_link), 333 CreateForumTopic (class in alogram.methods.create_chat_invite_link), 333 CreateForumTopic (class in alogram.methods.create_chat_invite_link), 333 CreateSpoin_request (class in alogram.types.chat_chat invite_link), 488 CreateLestype (alogram.types.melods.pupe.melods.pupe.goll.pull.pupe.message.pull.pupm.types.pupe.pre-checkout_query.PreCheckout_que		
currency (diogram.methods.send_invoice_SendInvoice attribute), 437  COP (diogram.enums.currency.Currency attribute), 475  copy() (diogram.enums.		
currency (aiogram.types.input_invoice_message_content.InputInvoiceMessage method), 564 copy() (aiogram.types.message.Message method), 565 copy_to() (aiogram.types.message.Message method), 198 copy_to() (aiogram.types.message), 331 copy_to() (aiogram.types.message), 332 copy_to() (aiogram.types.message), 333 copy_to() (aiogram.types.message), 331 copy_to() (aiogram.types.message), 331 copy_to() (aiogram.types.message), 331 copy_to() (aiogram.types.message), 331 copy_to() (aiogram.types.types.type) copy_to() (aiogram.types.types.types.type) copy_to() (aiogram.types.		currency (aiogram.methods.send_invoice.SendInvoice
COPY (aiogram.autils.keyboard.InlineKeyboardBuilder method), 564  copy() (aiogram.utils.keyboard.ReplyKeyboardBuilder method), 565  copy_to() (aiogram.tyls.keyboard.ReplyKeyboardBuilder method), 565  copy_to() (aiogram.types.message.Message method), 586  copy_to() (aiogram.types.message.Message.Message.methods.create.pel.methods.create.pel.methods.create.pel.pel.methods.create.pel.pel.pel.gel.methods.create.pel.pel.gel.gel.gel.gel.gel.gel.gel.gel.gel.g	ContentType (class in aiogram.enums.content_type),	attribute), 462
copy() (aiogram.utils.keyboard.InlineKeyboardBuilder method), 564 copy() (aiogram.utils.keyboard.ReplyKeyboardBuilder method), 565 copy=to() (aiogram.types.message.Message method), 198 copy=to() (aiogram.types.message.Message method), 198 copyMessage (class in aiogram.methods.copy_message), 319 corpect_option_id (aiogram.methods.copy_messages), 331 correct_option_id (aiogram.methods.send_poll.SendPollUSTON_EMO31 (aiogram.enums.message_entity_type_stribute), 495 correct_option_id (aiogram.methods.send_poll.SendPollUSTON_EMO31 (aiogram.enums.sticker_type.StickerType_attribute), 399 correct_option_id (aiogram.types.poll.Poll attribute), 212 country_code (aiogram.types.shipping_address.Shipping_add	473	$currency ({\it aiogram.types.input\_invoice\_message\_content.InputInvoiceMessage\_content$
method), 564  copy ( (aiogram.tuls.keyboard.ReplyKeyboardBuilder method), 565  copy_to() (aiogram.types.message.Message method), 198  copy_to() (aiogram.types.message.Message method), 198  copyMessage (class in aiogram.methods.copy_message), 329  copyMessages (class in aiogram.methods.copy_messages), 331  correct_option_id (aiogram.tupes.poll.SendPoll.	COP (aiogram.enums.currency.Currency attribute), 475	attribute), 273
copy() (aiogram.utils.keyboard.ReplyKeyboardBuilder method), 565 copy_tcO(_diogram.types.message.Message method), 198 CopyMessage (class in aiogram.methods.copy_message), 329 CopyMessages (class in aiogram.methods.copy_message), 329 CopyMessages (class in aiogram.methods.copy_message), 331 correct_option_id (aiogram.methods.send_poll.SendPoll*USTOM_EMOJ1 (aiogram.enums.message_entity_type.MessageEntityType attribute), 484 Correct_option_id (aiogram.types.poll.Poll attribute), 399 correct_option_id (aiogram.types.poll.Poll attribute), 212 country_code (aiogram.types.shipping_address.Shipping_Ackletcum_emoji_id (aiogram.types.message_entity_MessageEntity attribute), 312 country_codes (aiogram.types.giveaway attribute), 295 country_codes (aiogram.types.giveaway attribute), 135 CRC (aiogram.enums.currency.Currency attribute), 476 create_invite_link() (aiogram.types.chat_chat method), 42 create_start_link() (in module aiogram.methods.create_chat_imvite_link), 333 Creates_join_request	copy() (aiogram.utils.keyboard.InlineKeyboardBuilder	
currency (alogram.types.successful_payment.Successful_payment attribute), 297 coptret_option_id (alogram.types.send_send_poll.SendPollCustom_emoji_id (alogram.emums.currency), 475 coptret_option_id (alogram.types.poll.Poll attribute), 297 correct_option_id (alogram.types.poll.Poll attribute), 295 country_code (alogram.types.shipping_address.Shipping_Address.Shipping_Address.Shipping_address.Shi		
copy_to() (aiogram.types.message.Message method), 198 Copytessage (class in aiogram.methods.copy_message), 339 Correct_option_id (aiogram.methods.send_poll.SendPolCUSTOM_EM011 (aiogram.enums.message_entity_type.MessageEntityType attribute), 481 Correct_option_id (aiogram.types.poll.Poll attribute), 399 correct_option_id (aiogram.types.poll.Poll attribute), 212 country_code (aiogram.types.shipping_address.Shippi		
Currency (class in aiogram.enums.currency), 475 Custom_sage (class in aiogram.methods.copy_message), 329 CopyMessages (class in aiogram.methods.copy_message), 329 CopyMessages (class in aiogram.enums.message_entity_type_MessageEntityType_attribute), 481 Currency (class in aiogram.enums.message_entity_type_MessageEntityType_attribute), 483 Correct_option_id (aiogram.ethods.send_poll.SendPollCuSTOM_EMOII (aiogram.enums.reaction_type_type_ReactionTypeType_attribute), 483 Correct_option_id (aiogram.types.poll.Poll attribute), 206 Correct_option_id (aiogram.types.poll.Poll attribute), 212 Country_code (aiogram.types.shipping_address.Shipping_attribute), 212 Country_code (aiogram.types.shipping_address.Shipping_address.Shipping_attribute), 295 Country_code (aiogram.types.shipping_address.Shipping_address.Shipping_attribute), 206 Country_code (aiogram.types.shipping_address.Shipping_address.Shipping_address.Shipping_address.Shipping_address.Shipping_attribute), 312 Country_code (aiogram.types.shipping_address.Shipping		
COPyMessages (class in aiogram.methods.copy_message), aiogram.enthods.copy_messages), 331  Correct_option_id (aiogram.methods.send_poll.SendPoll*CUSTOM_EMO11 (aiogram.enums.reaction_type_type.ReactionTypeType attribute), 483  Correct_option_id (aiogram.types.spoll.Poll attribute), 476  Country_code (aiogram.types.shipping_address.Shipping Attribute), 212  Country_code (aiogram.types.shipping_address.Shipping Attribute), 295  Country_code (aiogram.types.giveaway.Giveaway attribute), 135  CRC (aiogram.enums.currency.Currency attribute), 476  Create_invite_link() (aiogram.types.shat.Chat method), 42  create_start_link() (aiogram.types.chat.Chat method), 42  create_start_link() (aiogram.types.chat.Chat attribute), 206  CreateChatInviteLink (class in aiogram.methods.create_chat_invite_link), 333  CreateSorumTopic (class in aiogram.methods.create_forum_topic), 334  CreateInvoiceLink (class in aiogram.methods.create_forum_topic), 334  CreateInvoiceLink (class in aiogram.methods.create_forum_topic), 334  CreateSorumTopic (class in aiogram.methods.create_invoice_link), 458  CreateNewStickerSet (class in aiogram.methods.create_new_sticker_set), 303  creates_join_request (aiogram.types.chat_member_administrator.ChatMemberOwner attribute), 333  creates_join_request (aiogram.types.chat_member_owner.ChatMemberOwner attribute), 333  creates_join_request (aiogram.types.chat_member_owner.ChatMemberOwner attribute), 333  creates_join_request (aiogram.types.chat_member_owner.ChatMemberOwner attribute), 333  creates_join_request (aiogram.types.background_type_fill.BackgroundIntribute), 476  D  Custom_enoji_id(aiogram.types.tater_t		
CopyMessages (class in aiogram.methods.copy_messages), 331		
CopyMessages (class in alogram.methods.copy_messages), 331 attribute), 483  Correct_option_id (alogram.methods.send_poll.SendPollCUSTOM_EMOJI (alogram.enums.sticker_type_StickerType attribute), 484  Correct_option_id (alogram.types.poll.Poll attribute), 212  Country_code (alogram.types.shipping_address.Shipping		
aiogram.methods.copy_messages), 331  correct_option_id(aiogram.methods.send_poll.SendPolCUSTOM_EMOJI (aiogram.enums.sticker_type.StickerType attribute), 484  correct_option_id(aiogram.types.poll.Poll attribute), 212  country_code (aiogram.types.shipping_address.ShippingAekdstxm_emoji_id(aiogram.types.message_entity.MessageEntity attribute), 312  country_codes (aiogram.types.giveaway.Giveaway attribute), 135  CRC (aiogram.enums.currency.Currency attribute), 476  create_invite_link() (aiogram.types.chat.Chat method), 42  create_start_link() (in module aiogram.utils.deep_linking), 591  CreateChatInviteLink (class in aiogram.methods.create_chat_invite_link), 333  CreateInvoiceLink (class in aiogram.methods.create_forum_topic), 334  CreateInvoiceLink (class in aiogram.methods.create_invoice_link), 458  CreateNewStickerSet (class in aiogram.methods.create_invoice_link), 458  CreateSpoin_request (aiogram.types.chat_member_administrator_custom_title.Set (aiogram.methods.create_chat_invite_link.Create CEttInvite_intem_diministrator.cy.Currency attribute), 476  attribute), 333  creates_join_request (aiogram.methods.edit_chat_invite_link.EditChatInvite_liteme_dimming (aiogram.types.background_type_fill.BackgroundInity_conditions_condi		
correct_option_id (aiogram.methods.send_poll.SendPolCUSTOM_EMOJI (aiogram.enums.sticker_type.StickerType attribute), 399 attribute), 484  correct_option_id (aiogram.types.poll.Poll attribute), 212  country_code (aiogram.types.shipping_address.ShippingAddsxcom_emoji_id (aiogram.types.message_entity.MessageEntity attribute), 295  country_codes (aiogram.types.giveaway.Giveaway attribute), 135  CRC (aiogram.enums.currency.Currency attribute), 476 create_invite_link() (aiogram.types.chat.Chat attribute), 215  CRC (aiogram.enums.currency.Currency attribute), 476 create_invite_link() (in module aiogram.mitls.deep_linking), 591  CreateChatInviteLink (class in aiogram.methods.create_chat_invite_link), 333  CreateForumTopic (class in aiogram.methods.create_invoice_link), 458  CreateInvoiceLink (class in aiogram.methods.create_invoice_link), 458  CreateInvoiceLink (class in aiogram.methods.create_invoice_link), 458  CreateSickerSet (class in aiogram.methods.create_invoice_link), 458  CreateSickerSet (class in aiogram.types.chat_member_owner.ChatMember.owner attribute), 303  creates_join_request (aiogram.methods.create_chat_invite_link.CreateCxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	· · · · · · · · · · · · · · · · · ·	
attribute), 399  correct_option_id (aiogram.types.poll.Poll attribute), 212  country_code (aiogram.types.shipping_address.ShippingAcdist.com_emoji_id (aiogram.types.message_entity.MessageEntity attribute), 295  country_codes (aiogram.types.giveaway.Giveaway attribute), 135  CRC (aiogram.enums.currency.Currency attribute), 476  create_invite_link() (aiogram.types.chat.Chat method), 42  create_start_link() (in module aiogram.mutils.deep_linking), 591  createChatInviteLink (class in aiogram.methods.create_chat_invite_link), 333  CreateForumTopic (class in aiogram.methods.create_forum_topic), 334  CreateInvoiceLink (class in aiogram.methods.create_invoice_link), 458  CreateNewStickerSet (class in aiogram.methods.create_invoice_link), 458  Creates_join_request (aiogram.types.chat_member_administrator.ChatMember.attribute), 333  creates_join_request (aiogram.types.chat_member_owner.ChatMemberOwner attribute), 333  creates_join_request (aiogram.methods.currency.currency attribute), 476  attribute), 333  creates_join_request (aiogram.types.background_type_fill.BackgroundInviveLink.EditChatInvite_ink.EditChatInvite_ink.EditChatInvite_ink.EditChatInvite_ink.EditChatInvite_ink.EditChatInvite_ink.EditChatInvite_ink.EditChatInvite_ink.EditChatInvite_ink.EditChatInvite_ink.EditChatInvite_iniming (aiogram.types.background_type_fill.BackgroundIntype_fill.Bac		
country_code (aiogram.types.shipping_address.ShippingAcdistom_emoji_id (aiogram.types.message_entity.MessageEntity attribute), 295  country_codes (aiogram.types.giveaway.Giveaway attribute), 135  CRC (aiogram.enums.currency.Currency attribute), 476  create_invite_link() (aiogram.types.chat.Chat method), 42  create_start_link() (in module aiogram.utils.deep_linking), 591  CreateChatInviteLink (class in aiogram.methods.create_chat_invite_link), 333  CreateForumTopic (class in aiogram.methods.create_forum_topic), 334  CreateInvoiceLink (class in aiogram.methods.create_invoice_link), 458  CreateNewStickerSet (class in aiogram.methods.create_invoice_link), 458  CreateSickerSet (class in aiogram.methods.create_new_sticker_set), 303  creates_join_request CustomEmoji (class in aiogram.utils.formatting), 586  (aiogram.methods.create_chat_invite_link.Create**  Custom_emoji_sticker_set_name (aiogram.types.chat_full_info.ChatFullInfo attribute), 476  custom_emoji_sticker_set_name (aiogram.types.chat_full_info.ChatFullInfo attribute), 476  custom_title (aiogram.types.chat_member_administrator.ChatMember.attribute), 412  custom_title (aiogram.types.chat_member_administrator.ChatMember.attribute), 476  custom_title (aiogr	attribute), 399	attribute), 484
attribute), 295  country_codes (aiogram.types.giveaway.Giveaway attribute), 135  CRC (aiogram.enums.currency.Currency attribute), 476 create_invite_link() (aiogram.types.chat.Chat method), 42  create_start_link() (in module aiogram.utils.deep_linking), 591  CreateChatInviteLink (class in aiogram.methods.create_chat_invite_link), 333  CreateForumTopic (class in aiogram.methods.create_forum_topic), 334  CreateInvoiceLink (class in aiogram.methods.create_inviveine_link), 458  CreateNewStickerSet (class in aiogram.methods.create_new_sticker_set), 303  creates_join_request (aiogram.methods.create_chat_invite_link. Create  (aiogram.methods.create_chat_invite_link. Create  CEXILIPIORED (All properties of the content of the		
country_codes (aiogram.types.giveaway.Giveaway attribute), 135  CRC (aiogram.enums.currency.Currency attribute), 476  create_invite_link() (aiogram.types.chat.Chat method), 42  create_start_link() (in module aiogram.mitls.deep_linking), 591  CreateChatInviteLink (class in aiogram.methods.create_chat_invite_link), 333  CreateForumTopic (class in aiogram.methods.create_invoice_link), 458  CreateInvoiceLink (class in aiogram.methods.create_chat_invite_link), 458  CreateNewStickerSet (class in aiogram.methods.create_chat_invite_link.Create  (aiogram.methods.create_chat_invite_link.Create  (aiogram.methods.create_chat_invite_link.Create  (aiogram.methods.create_chat_invite_link.Create  (CREATE (aiogram.types.chat_member_owner.ChatMemberOwner attribute), 333  creates_join_request  (aiogram.methods.create_chat_invite_link.Create  (aiogram.methods.create_chat_invite_link.Create  CREATE (aiogram.types.chat_member_owner.ChatMemberOwner attribute), 333  creates_join_request  (aiogram.methods.edit_chat_invite_link.EditChatlatifaxireLitkee_dimming (aiogram.types.background_type_fill.Background)  (aiogram.types.creaction_type_custom_emoji_Reaction attribute), 215  custom_emoji_id (aiogram.types.sticker.Sticker attribute), 279  custom_emoji_sticker_set_name  (aiogram.methods.get_custom_emoji_stickers.GetCustom_emoji_stickers.GetCustom_emoji_stickers.GetCustom_emoji_sticker_set_name  (aiogram.types.chat.Chat attribute), 36  custom_emoji_sticker_set_name  (aiogram.methods.set_chat_administrator_custom_title.Set_attribute), 412  custom_emoji_sticker_set_name  (aiogram.methods.set_chat_administrator_custom_title.Set_attribute), 101  custom_emoji_id (aiogram.types.chat_Chat attribute), 307  custom_emoji_sticker_set_name  (aiogram.methods.get_custom_emoji_stickers.GetCustom_emoji_sticker_set_name  (aiogram.methods.cetat_forum_iopic), 334  custom_emoji_id (aiogram.types.chat_Chat attribute), 307  custom_emoji_ids(aiogram.types.chat_futte), 307  custom_emoji_ide(aiogram.types.chat_futtente), 307  custom_emoji_ide(aiogram.types.chat_f	$\verb country_code   (a iogram. types. shipping\_address. Shipping\_ad$	Achdstrom_emoji_id (aiogram.types.message_entity.MessageEntity
tribute), 135  CRC (aiogram.enums.currency.Currency attribute), 476 create_invite_link() (aiogram.types.chat.Chat     method), 42     create_start_link() (in module         aiogram.mutils.deep_linking), 591  CreateChatInviteLink (class in         aiogram.methods.create_chat_invite_link),         333  CreateForumTopic (class in         aiogram.methods.create_forum_topic), 334  CreateInvoiceLink (class in         aiogram.methods.create_invoice_link), 458  CreateNewStickerSet (class in         aiogram.methods.create_new_sticker_set),         303  creates_join_request (aiogram.methods.edit_chat_invite_link.CreateCusting_invite_link.CreateCusting_invite_link.EditChatlatxire_lin		
CRC (aiogram.enums.currency.Currency attribute), 476 create_invite_link() (aiogram.types.chat.Chat		
create_invite_link() (aiogram.types.chat.Chat method), 42 create_start_link() (in module aiogram.utils.deep_linking), 591 custom_emoji_istickers.GetCustom_emoji_stickers.G	<i>**</i>	
create_start_link() (in module attribute), 307 createChatInviteLink (class in aiogram.methods.create_chat_invite_link), 333 CreateForumTopic (class in aiogram.methods.create_forum_topic), 334 CreateInvoiceLink (class in aiogram.methods.create_invoice_link), 458 CreateNewStickerSet (class in aiogram.methods.create_new_sticker_set), 303 creates_join_request (aiogram.methods.create_chat_invite_link.Create CEMI laving lainkenums.currency. Currency attribute), 476 attribute), 333 creates_join_request D custom_emoji_ids (aiogram.methods.get_custom_emoji_stickers.GetCu attribute), 307 custom_emoji_sticker_set_name (aiogram.types.chat.Chat attribute), 36 custom_emoji_sticker_set_name (aiogram.types.chat_full_info.ChatFullInfo custom_title (aiogram.methods.set_chat_administrator_custom_title.Set (aiogram.methods.create_invoice_link), 458 custom_title (aiogram.types.chat_member_administrator.ChatMember.custom_title), 101 custom_title (aiogram.types.chat_member_owner.ChatMemberOwner attribute), 103 creates_join_request (aiogram.methods.create_chat_invite_link.Create CEMI laving lainkenums.currency. Currency attribute), 476 attribute), 333 creates_join_request D (aiogram.methods.get_custom_emoji_sticker.set_name (aiogram.types.chat.Chat attribute), 36 custom_emoji_sticker_set_name (aiogram.types.chat_full_info.ChatFullInfo custom_title (aiogram.types.chat_member_administrator.ChatMember.cute), 101 custom_title (aiogram.types.chat_member_owner.ChatMemberOwner attribute), 103 creates_join_request (aiogram.methods.create_chat_invite_link.Create CEMI laving lainkenums.currency.Currency attribute), 476 attribute), 333 creates_join_request (aiogram.methods.dedit_chat_invite_link.EditChatlaving lainkenums.currency.Currency attribute), 476 attribute), 412 custom_title (aiogram.types.chat_member_owner.ChatMember.cute), 476 attribute), 103 custom_title (aiogram.types.chat_member_owner.ChatMember.cute), 476 attribute), 412 custom_title (aiogram.types.chat_member_owner.ChatMember.cute), 476 attribute), 412 custom_title (aiogram.types.		
create_start_link() (in module attribute), 307 createChatInviteLink (class in aiogram.methods.create_chat_invite_link), 333 CreateForumTopic (class in attribute), 36 CreateInvoiceLink (class in attribute), 36 CreateInvoiceLink (class in attribute), 412 aiogram.methods.create_invoice_link), 458 CreateNewStickerSet (class in aiogram.methods.create_new_sticker_set), 303 Creates_join_request (aiogram.methods.create_chat_invite_link.CreateCustluviveslainkenums.currency.Currency attribute), 476  (aiogram.methods.edit_chat_invite_link.EditChatUniveslainkenums.gana.types.background_type_fill.BackgroundInstrator_chatNember_dimining (aiogram.types.background_type_fill.BackgroundInstrator_chatNember_dimining (aiogram.types.background_type_fill.Background_type_fi		
aiogram.utils.deep_linking), 591custom_emoji_sticker_set_nameCreateChatInviteLink(classin(aiogram.types.chat.Chat attribute), 36aiogram.methods.create_chat_invite_link), 333custom_emoji_sticker_set_namecustom_emoji_sticker_set_name333(aiogram.types.chat_full_info.ChatFullInfoCreateForumTopic(classinattribute), 60aiogram.methods.create_forum_topic), 334custom_title (aiogram.methods.set_chat_administrator_custom_title.Set_in attribute), 412CreateInvoiceLink(classinattribute), 412aiogram.methods.create_invoice_link), 458custom_title (aiogram.types.chat_member_administrator.ChatMember.attribute), 101CreateNewStickerSet(classinattribute), 101aiogram.methods.create_new_sticker_set), 303custom_title (aiogram.types.chat_member_owner.ChatMemberOwner_attribute), 103creates_join_requestCustomEmoji (class in aiogram.utils.formatting), 586(aiogram.methods.create_chat_invite_link.CreateClast Invite_dinkenums.currency.Currency attribute), 476D(aiogram.methods.edit_chat_invite_link.EditChatlast Invite_link.EditChatlast Invite_		
CreateChatInviteLink (class in aiogram.methods.create_chat_invite_link), 333 CreateForumTopic (class in attribute), 60 aiogram.methods.create_forum_topic), 334 CreateInvoiceLink (class in attribute), 412 aiogram.methods.create_invoice_link), 458 CreateNewStickerSet (class in aiogram.methods.create_new_sticker_set), 303 creates_join_request (aiogram.methods.create_chat_invite_link.Create (aiogram.methods.create_chat_invite_link.EditChatlmaitkLittleme_dimming (aiogram.types.background_type_fill.BackgroundInterval attribute), 26 custom_title (aiogram.types.chat_mattribute), 36 custom_title (aiogram.types.chat_member_administrator.ChatMember.custom_title.Set in attribute), 101 custom_title (aiogram.types.chat_member_administrator.ChatMember.custom_title (aiogram.types.chat_member_owner.ChatMember.custom_title (aiogram.types.chat_member_owner		
aiogram.methods.create_chat_invite_link), 333  CreateForumTopic (class in attribute), 60 aiogram.methods.create_forum_topic), 334 CreateInvoiceLink (class in aiogram.methods.create_invoice_link), 458 CreateNewStickerSet (class in aiogram.methods.create_new_sticker_set), 303 creates_join_request (aiogram.methods.create_chat_invite_link.CreateCust lawinglainkenums.currency.Currency attribute), 476 attribute), 333 creates_join_request (aiogram.methods.create_chat_invite_link.EditChatlawinglainkenums(aiogram.types.background_type_fill.BackgroundInvite_link.EditChatlawinglainkenuming (aiogram.types.background_type_fill.BackgroundInvite_link.EditChatlawinglainkenuming (aiogram.types.background_type_fill.BackgroundInvite_link.EditChatlawinglainkenuming (aiogram.types.background_type_fill.BackgroundInvite_link.EditChatlawinglainkenuming (aiogram.types.background_type_fill.BackgroundInvite_link.EditChatlawinglainkenuming (aiogram.types.background_type_fill.BackgroundInvite_link.EditChatlawinglainkenums.currency.Currency.attribute_link.EditChatlawinglainkenuming (aiogram.types.background_type_fill.BackgroundInvite_link.EditChatlawinglainkenuming)  custom_title(aiogram.methods.set_chat_administrator_custom_title.Set_attribute), 412 custom_title(aiogram.methods.set_chat_administrator_custom_title.Set_attribute), 412 custom_title(aiogram.types.chat_member_administrator.ChatMember.attribute), 101 custom_title(aiogram.types.chat_member_administrator.ChatMember.attribute), 103 custom_title(aiogram.types.chat_member_administrator_custom_title.Set_attribute), 103 custom_title(aiogram.types.chat_member_administrator_custom_title.Set_attribute), 103 custom_title(aiogram.types.chat_member_administrator.ChatMember.attribute), 103 custom_title(aiogram.types.chat_member_administrator.ChatMember.attribute), 103 custom_title(aiogram.types.chat_member_administrator.ChatMember.attribute), 103 custom_title(aiogram.types.chat_member_administrator.ChatMember.attribute), 103 custom_title(aiogram.types.chat_member_administrator.ChatMember.attr	ž	
(aiogram.types.chat_full_info.ChatFullInfo  CreateForumTopic (class in attribute), 60  aiogram.methods.create_forum_topic), 334  CreateInvoiceLink (class in attribute), 412  aiogram.methods.create_invoice_link), 458  CreateNewStickerSet (class in aiogram.types.chat_member_administrator.ChatMember.title (aiogram.types.chat_member_administrator.ChatMember.title), 101  aiogram.methods.create_new_sticker_set), 303  creates_join_request (aiogram.methods.create_chat_invite_link.CreateClass in aiogram.types.chat_member_owner.ChatMemberOwner attribute), 103  creates_join_request (aiogram.methods.create_chat_invite_link.CreateClass in aiogram.utils.formatting), 586  (attribute), 333  creates_join_request D  (aiogram.methods.edit_chat_invite_link.EditChatlasiteLinkeme_dimming (aiogram.types.background_type_fill.BackgroundIndex.edit_chat_invite_link.EditChatlasiteLinkeme_dimming (aiogram.types.background_type_fill.BackgroundIndex.edit_chat_invite_link.EditChatlasiteLinkeme_dimming (aiogram.types.background_type_fill.BackgroundIndex.edit_chat_invite_link.EditChatlasiteLinkeme_dimming (aiogram.types.background_type_fill.BackgroundIndex.edit_chat_invite_link.EditChatlasiteLinkeme_dimming (aiogram.types.background_type_fill.BackgroundIndex.edit_chat_invite_link.EditChatlasiteLinkeme_dimming (aiogram.types.background_type_fill.BackgroundIndex.edit_chat_invite_link.EditChatlasiteLinkeme_dimming (aiogram.types.background_type_fill.BackgroundIndex.edit_chat_invite_link.EditChatlasiteLinkeme_dimming_aiogram.edit_chat_invite_linkeme_dimming_aiogram.edit_chat_invite_linkeme_dimming_aiogram.edit_chat_invite_linkeme_dimming_aiogram.edit_chat_invite_linkeme_dimming_aiogram.edit_chat_invite_linkeme_dimming_aiogram.edit_chat_invite_linkeme_dimming_aiogram.edit_chat_invite_linkeme_dimming_aiogram.edit_chat_invite_linkeme_dimming_aiogram.edit_chat_invite_linkeme_dimming_aiogram.edit_chat_invite_linkeme_dimming_aiogram.edit_chat_invite_linkeme_dimming_aiogram.edit_chat_invite_linkeme_dimming_aiogram.edit_chat_invite_linkeme_dimming_aiogram.	· ·	
CreateInvoiceLink (class in attribute), 60  CreateInvoiceLink (class in attribute), 412  aiogram.methods.create_invoice_link), 458  CreateNewStickerSet (class in aiogram.methods.create_new_sticker_set), 303  creates_join_request (aiogram.methods.create_chat_invite_link.CreateClast laving lankenums.currency.Currency attribute), 476  attribute), 333  creates_join_request (aiogram.methods.create_chat_invite_link.CreateClast laving lankenums.currency.Currency attribute), 476  attribute), 333  creates_join_request (aiogram.methods.edit_chat_invite_link.EditChatlaving lankenums (aiogram.types.background_type_fill.Background)  CreateForumTopic (class in attribute), 60  custom_title (aiogram.methods.set_chat_administrator_custom_title.Set attribute), 412  custom_title (aiogram.types.chat_member_administrator.ChatMember.custom_title.Set attribute), 101  custom_title (aiogram.types.chat_member_administrator.ChatMember.custom_title.Set attribute), 103  custom_title (aiogram.types.chat_member_administrator.ChatMember.custom_title (aiogram.types.chat_me		
createInvoiceLink (class in attribute), 412 createNewStickerSet (class in attribute), 101 creates_join_request (aiogram.methods.create_chat_invite_link.CreateCINK) (aiogram.methods.create_chat_invite_link.CreateCINK) (aiogram.methods.create_chat_invite_link.EditChatlaxitkLinkeme_dimming (aiogram.types.background_type_fill.BackgroundInterpret)  custom_title(aiogram.methods.set_chat_administrator_custom_title.Set_in attribute), 412  custom_title(aiogram.types.chat_member_administrator.ChatMember.Coustom_title(aiogram.types.chat_member_administrator.ChatMember.Coustom_title(aiogram.types.chat_member_administrator.ChatMember.Coustom_title(aiogram.types.chat_member_administrator.ChatMember.Coustom_title(aiogram.types.chat_member_administrator.ChatMember.Coustom_title(aiogram.types.chat_member_administrator.ChatMember.Coustom_title(aiogram.types.chat_member_administrator.ChatMember.Coustom_title(aiogram.types.chat_member_administrator.ChatMember.Coustom_title(aiogram.types.chat_member_administrator.ChatMember.Coustom_title(aiogram.types.chat_member_administrator.ChatMember.Coustom_title(aiogram.types.chat_member_administrator.ChatMember.Coustom_title(aiogram.types.chat_member_administrator.ChatMember.Coustom_title(aiogram.types.chat_member_administrator.ChatMember.Coustom_title(aiogram.types.chat_member_administrator.ChatMember.Coustom_title(aiogram.types.chat_member_administrator.ChatMember.Coustom_title(aiogram.types.chat_member_administrator.ChatMember.Coustom_title(aiogram.types.chat_member_administrator.ChatMember.Coustom_title(aiogram.types.chat_member_administrator.ChatMember.Coustom_title(aiogram.types.chat_member_administrator.ChatMember.Coustom_title(aiogram.types.chat_member_administrator.ChatMember.Coustom_title(aiogram.types.chat_member_administrator.ChatMember.Coustom_title(aiogram.types.chat_member_administrator.ChatMember.Coustom_title(aiogram.types.chat_member_administrator.ChatMember.Coustom_title(aiogram.types.chat_member_administrator.ChatMember.Coustom_title(aiogram.types.chat_member_administrato		. 0 11 - 0
CreateInvoiceLink (class in attribute), 412     aiogram.methods.create_invoice_link), 458 CreateNewStickerSet (class in attribute), 101     aiogram.methods.create_new_sticker_set),		
createNewStickerSet (class in attribute), 101 aiogram.methods.create_new_sticker_set), 303 creates_join_request (class in attribute), 103 creates_join_request (class in aiogram.methods.create_chat_invite_link.Create CTMst Laving Linkenums.currency.Currency attribute), 476 attribute), 333 creates_join_request D (aiogram.methods.edit_chat_invite_link.EditChat Laving Linkenums] (aiogram.types.background_type_fill.Background)		
CreateNewStickerSet (class in attribute), 101  aiogram.methods.create_new_sticker_set), custom_title(aiogram.types.chat_member_owner.ChatMemberOwner 303  creates_join_request CustomEmoji (class in aiogram.utils.formatting), 586  (aiogram.methods.create_chat_invite_link.Create CustInvite_linkenums.currency.Currency attribute), 476  attribute), 333  creates_join_request D  (aiogram.methods.edit_chat_invite_link.EditChatInvite_link.EditChatInvite_link.invite_link.EditChatInvite_link.eme_dimming(aiogram.types.background_type_fill.BackgroundInvite_link.eme_dimming(aiogram.types.background_type_fill.BackgroundInvite_link.eme_dimming(aiogram.types.background_type_fill.BackgroundInvite_link.eme_dimming(aiogram.types.background_type_fill.BackgroundInvite_link.eme_dimming(aiogram.types.background_type_fill.BackgroundInvite_link.eme_dimming(aiogram.types.background_type_fill.BackgroundInvite_link.eme_dimming(aiogram.types.background_type_fill.BackgroundInvite_link.eme_dimming(aiogram.types.background_type_fill.BackgroundInvite_link.eme_dimming(aiogram.types.background_type_fill.BackgroundInvite_link.eme_dimming(aiogram.types.background_type_fill.BackgroundInvite_link.eme_dimming(aiogram.types.background_type_fill.BackgroundInvite_link.eme_dimming(aiogram.types.background_type_fill.BackgroundInvite_dimming(aiogram.types.background_type_fill.Background_type_fill.Background_type_fill.Background_type_fill.Background_type_fill.Background_type_fill.Background_type_fill.Background_type_fill.Background_type_fill.Background_type_fill.Background_type_fill.Background_fill.Backg	· ·	
aiogram.methods.create_new_sticker_set), custom_title(aiogram.types.chat_member_owner.ChatMemberOwner 303 attribute), 103  creates_join_request CustomEmoji (class in aiogram.utils.formatting), 586		The second secon
attribute), 103  creates_join_request	*	
creates_join_request CustomEmoji (class in aiogram.utils.formatting), 586		
(aiogram.methods.create_chat_invite_link.Create@thtlavivelainkenums.currency.Currency attribute), 476 attribute), 333 creates_join_request		
attribute), 333  creates_join_request  D  (aiogram.methods.edit_chat_invite_link.EditChatlaviteLinkeme_dimming(aiogram.types.background_type_fill.Background)		
creates_join_request  (aiogram.methods.edit_chat_invite_link.EditChatlayiteLinkeme_dimming(aiogram.types.background_type_fill.Background]	·	Commence of the second section of the second
(aiogram.methods.edit_chat_invite_link.EditChatlayiteLinkeme_dimming(aiogram.types.background_type_fill.BackgroundI		D
	attribute), 341	

attribute), 24	w <b>allejlæjtæ:B</b> ack <b>gniogndiflytygWstHæspag</b> e.Message method), 203
DART (aiogram.enums.dice_emoji.DiceEmoji attribute), 478	DELETE_CHAT_PHOTO (aiogram.enums.content_type.ContentType attribute), 474
DART (aiogram.types.dice.DiceEmoji attribute), 127	delete_chat_photo (aiogram.types.message.Message
DATA (aiogram.enums.passport_element_error_type.Passp	portElement <b>EtmibTiyp</b> ); 160
attribute), 482	delete_from_set() (aiogram.types.sticker.Sticker
data (aiogram.types.callback_query.CallbackQuery at-	· · · · · · · · · · · · · · · · · · ·
tribute), 34	<pre>delete_message() (aiogram.types.chat.Chat method),</pre>
data(aiogram.types.encrypted_credentials.EncryptedCre	
attribute), 281	<pre>delete_photo() (aiogram.types.chat.Chat method), 49</pre>
data(aiogram.types.encrypted_passport_element.Encryp	
attribute), 282	(aiogram.types.message.Message method),
data (aiogram.types.passport_data.PassportData at-	
tribute), 283	<pre>delete_sticker_set() (aiogram.types.chat.Chat</pre>
data (aiogram.types.web_app_data.WebAppData at-	· · · · · · · · · · · · · · · · · · ·
tribute), 228	DeleteChatPhoto (class in
data_hash(aiogram.types.passport_element_error_data	· ·
attribute), 285	DeleteChatStickerSet (class in
date(aiogram.types.business_connection.BusinessConne	· · · · · · · · · · · · · · · · · · ·
attribute), 31	337
date (aiogram.types.chat_join_request.ChatJoinRequest	
attribute), 61	(aiogram.enums.update_type.UpdateType
date(aiogram.types.chat_member_updated.ChatMember	
attribute), 106	deleted_business_messages
date (aiogram.types.inaccessible_message.InaccessibleM	
attribute), 137	299
date (aiogram.types.message.Message attribute), 158	DeleteForumTopic (class in
	riginChannehiogram.methods.delete_forum_topic), 338
attribute), 207	DeleteMessage (class in
attribute), 207 date(aiogram.types.message_origin_chat.MessageOrigin	DeleteMessage (class in mChat aiogram.methods.delete_message), 434
attribute), 207 date(aiogram.types.message_origin_chat.MessageOrigin_attribute), 208	DeleteMessage (class in nChat aiogram.methods.delete_message), 434 DeleteMessages (class in
attribute), 207 date(aiogram.types.message_origin_chat.MessageOrigin_attribute), 208 date(aiogram.types.message_origin_hidden_user.Message_origin_hidden_user.Message_origin_hidden_user.Message_origin_hidden_user.Message_origin_hidden_user.Message_origin_hidden_user.Message_origin_hidden_user.Message_origin_hidden_user.Message_origin_hidden_user.Message_origin_hidden_user.Message_origin_origin_hidden_user.Message_origin_hid	DeleteMessage (class in mchat aiogram.methods.delete_message), 434  DeleteMessages (class in geOriginHiddingVsmr.methods.delete_messages), 436
attribute), 207 date (aiogram.types.message_origin_chat.MessageOrigin_attribute), 208 date (aiogram.types.message_origin_hidden_user.Messagattribute), 208	DeleteMessage (class in nChat aiogram.methods.delete_message), 434  DeleteMessages (class in geOriginHiddingNsmr.methods.delete_messages), 436  DeleteMyCommands (class in
attribute), 207 date (aiogram.types.message_origin_chat.MessageOrigin_attribute), 208 date (aiogram.types.message_origin_hidden_user.Message_origin_bidden_user.Message_origin_user.MessageOrigin_attribute), 208	DeleteMessage (class in mchat aiogram.methods.delete_message), 434  DeleteMessages (class in geOriginHiddiomy/ser.methods.delete_messages), 436  DeleteMyCommands (class in muser aiogram.methods.delete_my_commands), 339
attribute), 207  date (aiogram.types.message_origin_chat.MessageOrigin_attribute), 208  date (aiogram.types.message_origin_hidden_user.Message_attribute), 208  date (aiogram.types.message_origin_user.MessageOrigin_attribute), 209  date (aiogram.types.message_reaction_count_updated.M	DeleteMessage (class in mchat aiogram.methods.delete_message), 434  DeleteMessages (class in geOriginHiddiongl/smr.methods.delete_messages), 436  DeleteMyCommands (class in muser aiogram.methods.delete_my_commands), 339  Messaged Restributement by Stated (class in
attribute), 207  date (aiogram.types.message_origin_chat.MessageOrigin_attribute), 208  date (aiogram.types.message_origin_hidden_user.Message_attribute), 208  date (aiogram.types.message_origin_user.MessageOrigin_attribute), 209  date (aiogram.types.message_reaction_count_updated.M	DeleteMessage (class in mchat aiogram.methods.delete_message), 434  DeleteMessages (class in geOriginHiddiomy/ser.methods.delete_messages), 436  DeleteMyCommands (class in muser aiogram.methods.delete_my_commands), 339
attribute), 207  date (aiogram.types.message_origin_chat.MessageOrigin_attribute), 208  date (aiogram.types.message_origin_hidden_user.Message_attribute), 208  date (aiogram.types.message_origin_user.MessageOrigin_attribute), 209  date (aiogram.types.message_reaction_count_updated.M	DeleteMessage (class in mchat aiogram.methods.delete_message), 434  DeleteMessages (class in geOriginHiddingVism.methods.delete_messages), 436  DeleteMyCommands (class in muser aiogram.methods.delete_my_commands), 339  Jes DeleteMyComFit bynSeted (class in aiogram.methods.delete_sticker_from_set),
attribute), 207  date (aiogram.types.message_origin_chat.MessageOrigin_attribute), 208  date (aiogram.types.message_origin_hidden_user.Message_attribute), 208  date (aiogram.types.message_origin_user.MessageOrigin_attribute), 209  date (aiogram.types.message_reaction_count_updated.Mattribute), 210	DeleteMessage (class in mchat aiogram.methods.delete_message), 434  DeleteMessages (class in geOriginHiddingVism.methods.delete_messages), 436  DeleteMyCommands (class in muser aiogram.methods.delete_my_commands), 339  Jes DeleteMyComFit bynSeted (class in aiogram.methods.delete_sticker_from_set),
attribute), 207  date (aiogram.types.message_origin_chat.MessageOrigin_attribute), 208  date (aiogram.types.message_origin_hidden_user.Message_attribute), 208  date (aiogram.types.message_origin_user.MessageOrigin_attribute), 209  date (aiogram.types.message_reaction_count_updated.Message), 210  date (aiogram.types.message_reaction_updated.Message)	DeleteMessage (class in mchat aiogram.methods.delete_message), 434  DeleteMessages (class in geOriginHiddingVism.methods.delete_messages), 436  DeleteMyCommands (class in muser aiogram.methods.delete_my_commands), 339  VesDedanceStivoKenFirbynSated (class in aiogram.methods.delete_sticker_from_set),  ReactionUpAnted
attribute), 207  date (aiogram.types.message_origin_chat.MessageOrigin_attribute), 208  date (aiogram.types.message_origin_hidden_user.Message_attribute), 208  date (aiogram.types.message_origin_user.MessageOrigin_attribute), 209  date (aiogram.types.message_reaction_count_updated.Mattribute), 210  date (aiogram.types.message_reaction_updated.Message_attribute), 210	DeleteMessage (class in mchat aiogram.methods.delete_message), 434  DeleteMessages (class in geOriginHiddingNam.methods.delete_messages), 436  DeleteMyCommands (class in muser aiogram.methods.delete_my_commands), 339  MessagaRaestivakenFrbynSeted (class in aiogram.methods.delete_sticker_from_set),  ReactionUpdated  DeleteStickerSet (class in aiogram.methods.delete_sticker_from_set), 306
attribute), 207  date (aiogram.types.message_origin_chat.MessageOrigin_attribute), 208  date (aiogram.types.message_origin_hidden_user.Message_attribute), 208  date (aiogram.types.message_origin_user.MessageOrigin_attribute), 209  date (aiogram.types.message_reaction_count_updated.Matribute), 210  date (aiogram.types.message_reaction_updated.Message_attribute), 210  day (aiogram.types.birthdate.Birthdate attribute), 25	DeleteMessage (class in mchat aiogram.methods.delete_message), 434  DeleteMessages (class in geOriginHiddingNam.methods.delete_messages), 436  DeleteMyCommands (class in muser aiogram.methods.delete_my_commands), 339  MessagaRaestivakenFrbynSeted (class in aiogram.methods.delete_sticker_from_set),  ReactionUpdated  DeleteStickerSet (class in aiogram.methods.delete_sticker_from_set), 306
attribute), 207  date (aiogram.types.message_origin_chat.MessageOrigin_attribute), 208  date (aiogram.types.message_origin_hidden_user.Message_attribute), 208  date (aiogram.types.message_origin_user.MessageOrigin_attribute), 209  date (aiogram.types.message_reaction_count_updated.Mattribute), 210  date (aiogram.types.message_reaction_updated.Message_attribute), 210  day (aiogram.types.message_reaction_updated.Message_attribute), 25  decline() (aiogram.types.birthdate.Birthdate attribute), 25  decline() (aiogram.types.chat_join_request.ChatJoinRemethod), 62	DeleteMessage (class in action of the content of th
attribute), 207  date (aiogram.types.message_origin_chat.MessageOrigin_attribute), 208  date (aiogram.types.message_origin_hidden_user.Message_attribute), 208  date (aiogram.types.message_origin_user.MessageOrigin_attribute), 209  date (aiogram.types.message_reaction_count_updated.Mattribute), 210  date (aiogram.types.message_reaction_updated.Message_attribute), 210  day (aiogram.types.message_reaction_updated.Message_attribute), 25  decline() (aiogram.types.birthdate.Birthdate attribute), 25  decline() (aiogram.types.chat_join_request.ChatJoinRemethod), 62	DeleteMessage (class in mchat aiogram.methods.delete_message), 434  DeleteMessages (class in geOriginHiddingVism.methods.delete_messages), 436  DeleteMyCommands (class in muser aiogram.methods.delete_my_commands), 339  VesDedareStionKerFirbynSated (class in aiogram.methods.delete_sticker_from_set),  ReactionUparted  DeleteStickerSet (class in aiogram.methods.delete_sticker_set), 306  equivaleteWebhook (class in aiogram.methods.delete_webhook), 464
attribute), 207  date (aiogram.types.message_origin_chat.MessageOrigin_attribute), 208  date (aiogram.types.message_origin_hidden_user.Message_attribute), 208  date (aiogram.types.message_origin_user.MessageOrigin_attribute), 209  date (aiogram.types.message_reaction_count_updated.Mattribute), 210  date (aiogram.types.message_reaction_updated.Message_attribute), 210  day (aiogram.types.message_reaction_updated.Message_attribute), 210  day (aiogram.types.birthdate.Birthdate attribute), 25  decline() (aiogram.types.chat_join_request.ChatJoinRemethod), 62  DeclineChatJoinRequest (class in	DeleteMessage (class in nechat aiogram.methods.delete_message), 434  DeleteMessages (class in geOriginHiddioglysm.methods.delete_messages), 436  DeleteMyCommands (class in nechals delete_my_commands), 339  Jes DeleteMyCommands (class in aiogram.methods.delete_my_commands), 339  Jes DeleteMyCommands (class in aiogram.methods.delete_sticker_from_set),  ReactionUpAnted  DeleteStickerSet (class in aiogram.methods.delete_sticker_set), 306  equival eteWebhook (class in aiogram.methods.delete_webhook), 464  description (aiogram.methods.create_invoice_link.CreateInvoiceLink
attribute), 207  date (aiogram.types.message_origin_chat.MessageOrigin_attribute), 208  date (aiogram.types.message_origin_hidden_user.Message_attribute), 208  date (aiogram.types.message_origin_user.MessageOrigin_attribute), 209  date (aiogram.types.message_reaction_count_updated.Maattribute), 210  date (aiogram.types.message_reaction_updated.Message_attribute), 210  day (aiogram.types.message_reaction_updated.Message_attribute), 210  day (aiogram.types.birthdate.Birthdate attribute), 25  decline() (aiogram.types.chat_join_request.ChatJoinRemethod), 62  DeclineChatJoinRequest (class in aiogram.methods.decline_chat_join_request),	DeleteMessage (class in nechat aiogram.methods.delete_message), 434  DeleteMessages (class in geOriginHiddingNism.methods.delete_messages), 436  DeleteMyCommands (class in necket aiogram.methods.delete_my_commands), 339  MessageRiestivokentropseted (class in aiogram.methods.delete_sticker_from_set),  ReactionUpdiviod  DeleteStickerSet (class in aiogram.methods.delete_sticker_set), 306  equivaleteWebhook (class in aiogram.methods.delete_sticker_set), 306  equivaleteWebhook (class in aiogram.methods.delete_webhook), 464  description (aiogram.methods.create_invoice_link.CreateInvoiceLink attribute), 459  description (aiogram.methods.send_invoice.SendInvoice
attribute), 207  date (aiogram.types.message_origin_chat.MessageOrigin_attribute), 208  date (aiogram.types.message_origin_hidden_user.Message_attribute), 208  date (aiogram.types.message_origin_user.MessageOrigin_attribute), 209  date (aiogram.types.message_reaction_count_updated.Maattribute), 210  date (aiogram.types.message_reaction_updated.Message_attribute), 210  day (aiogram.types.message_reaction_updated.Message_attribute), 210  day (aiogram.types.birthdate.Birthdate attribute), 25  decline() (aiogram.types.chat_join_request.ChatJoinRemethod), 62  DeclineChatJoinRequest (class in aiogram.methods.decline_chat_join_request), 335	DeleteMessage (class in nechat aiogram.methods.delete_message), 434  DeleteMessages (class in geOriginHiddingNism.methods.delete_messages), 436  DeleteMyCommands (class in new nechation diogram.methods.delete_my_commands), 339  Messagaragaragaragaragaragaragaragaragarag
attribute), 207  date (aiogram.types.message_origin_chat.MessageOrigin_attribute), 208  date (aiogram.types.message_origin_hidden_user.Message_attribute), 208  date (aiogram.types.message_origin_user.MessageOrigin_attribute), 209  date (aiogram.types.message_reaction_count_updated.Maattribute), 210  date (aiogram.types.message_reaction_updated.Message_attribute), 210  day (aiogram.types.message_reaction_updated.Message_attribute), 210  day (aiogram.types.birthdate.Birthdate attribute), 25  decline() (aiogram.types.chat_join_request.ChatJoinRemethod), 62  DeclineChatJoinRequest(classin_aiogram.methods.decline_chat_join_request), 335  decode_payload() (in_module	DeleteMessage (class in nechat aiogram.methods.delete_message), 434  DeleteMessages (class in geOriginHiddingNam.methods.delete_messages), 436  DeleteMyCommands (class in new nechaes delete_my_commands), 339  Vestagatagatagatagatagatagatagatagatagatag
attribute), 207  date (aiogram.types.message_origin_chat.MessageOrigin_attribute), 208  date (aiogram.types.message_origin_hidden_user.Message_attribute), 208  date (aiogram.types.message_origin_user.MessageOrigin_attribute), 209  date (aiogram.types.message_reaction_count_updated.Message_attribute), 210  date (aiogram.types.message_reaction_updated.Message_attribute), 210  day (aiogram.types.message_reaction_updated.Message_attribute), 210  day (aiogram.types.birthdate.Birthdate attribute), 25  decline() (aiogram.types.chat_join_request.ChatJoinRemethod), 62  DeclineChatJoinRequest(classin_aiogram.methods.decline_chat_join_request), 335  decode_payload()(inmodule_aiogram.utils.deep_linking), 591	DeleteMessage (class in nechat aiogram.methods.delete_message), 434  DeleteMessages (class in geOriginHiddingNam.methods.delete_messages), 436  DeleteMyCommands (class in new nechaes delete_my_commands), 339  Vestagatagatagatagatagatagatagatagatagatag
attribute), 207  date (aiogram.types.message_origin_chat.MessageOrigin_attribute), 208  date (aiogram.types.message_origin_hidden_user.Message_attribute), 208  date (aiogram.types.message_origin_user.MessageOrigin_attribute), 209  date (aiogram.types.message_reaction_count_updated.Message_attribute), 210  date (aiogram.types.message_reaction_updated.Message_attribute), 210  day (aiogram.types.message_reaction_updated.Message_attribute), 210  day (aiogram.types.birthdate.Birthdate attribute), 25  decline() (aiogram.types.chat_join_request.ChatJoinRemethod), 62  DeclineChatJoinRequest (class in aiogram.methods.decline_chat_join_request), 335  decode_payload() (in module_aiogram.utils.deep_linking), 591  DEFAULT (aiogram.enums.bot_command_scope_type.Boto	DeleteMessage (class in mchat aiogram.methods.delete_message), 434  DeleteMessages (class in geOriginHiddingVan:methods.delete_messages), 436  DeleteMyCommands (class in muser aiogram.methods.delete_my_commands), 339  desDedicteStiveKenFrbinStated (class in aiogram.methods.delete_sticker_from_set),  ReactionUpdated  DeleteStickerSet (class in aiogram.methods.delete_sticker_set), 306  equentetleteWebhook (class in aiogram.methods.delete_sticker_set), 306  equentetleteWebhook (class in aiogram.methods.delete_webhook), 464  description (aiogram.methods.create_invoice_link.CreateInvoiceLink attribute), 459  description (aiogram.methods.send_invoice.SendInvoice attribute), 462  description (aiogram.methods.set_chat_description.SetChatDescription CommandScappatTypte), 413  description (aiogram.methods.set_my_description.SetMyDescription
attribute), 207  date (aiogram.types.message_origin_chat.MessageOrigin_attribute), 208  date (aiogram.types.message_origin_hidden_user.Message_attribute), 208  date (aiogram.types.message_origin_user.MessageOrigin_attribute), 209  date (aiogram.types.message_reaction_count_updated.Maattribute), 210  date (aiogram.types.message_reaction_updated.Message_attribute), 210  day (aiogram.types.message_reaction_updated.Message_attribute), 210  day (aiogram.types.birthdate.Birthdate attribute), 25  decline() (aiogram.types.chat_join_request.ChatJoinRemethod), 62  DeclineChatJoinRequest (class in aiogram.methods.decline_chat_join_request), 335  decode_payload() (in module_aiogram.utils.deep_linking), 591  DEFAULT (aiogram.enums.bot_command_scope_type.Boto_attribute), 471	DeleteMessage (class in mchat aiogram.methods.delete_message), 434  DeleteMessages (class in geOriginHiddingVan:methods.delete_messages), 436  DeleteMyCommands (class in muser aiogram.methods.delete_my_commands), 339  desDedicteStiveKenFrbinStated (class in aiogram.methods.delete_sticker_from_set),  ReactionUpdated  DeleteStickerSet (class in aiogram.methods.delete_sticker_set), 306  equentetleteWebhook (class in aiogram.methods.delete_sticker_set), 306  equentetleteWebhook (class in aiogram.methods.delete_webhook), 464  description (aiogram.methods.create_invoice_link.CreateInvoiceLink attribute), 459  description (aiogram.methods.send_invoice.SendInvoice attribute), 462  description (aiogram.methods.set_chat_description.SetChatDescription CommandScappatTypte), 413  description (aiogram.methods.set_my_description.SetMyDescription
attribute), 207  date (aiogram.types.message_origin_chat.MessageOrigin_attribute), 208  date (aiogram.types.message_origin_hidden_user.Message_attribute), 208  date (aiogram.types.message_origin_user.MessageOrigin_attribute), 209  date (aiogram.types.message_reaction_count_updated.Maattribute), 210  date (aiogram.types.message_reaction_updated.Message_attribute), 210  day (aiogram.types.message_reaction_updated.Message_attribute), 210  day (aiogram.types.birthdate.Birthdate attribute), 25  decline() (aiogram.types.chat_join_request.ChatJoinRemethod), 62  DeclineChatJoinRequest (class in aiogram.methods.decline_chat_join_request), 335  decode_payload() (in module_aiogram.utils.deep_linking), 591  DEFAULT (aiogram.enums.bot_command_scope_type.BotOattribute), 471  DEFAULT (aiogram.enums.menu_button_type.MenuButton_type.Me	DeleteMessage (class in mchat aiogram.methods.delete_message), 434  DeleteMessages (class in geOriginHiddionf/smr.methods.delete_messages), 436  DeleteMyCommands (class in muser aiogram.methods.delete_my_commands), 339  desDedataeStivoKentirofpSeted (class in aiogram.methods.delete_sticker_from_set),  ReactionUpdated  DeleteStickerSet (class in aiogram.methods.delete_sticker_set), 306  equenteteWebhook (class in aiogram.methods.delete_webhook), 464  description(aiogram.methods.delete_webhook), 464  description(aiogram.methods.create_invoice_link.CreateInvoiceLink attribute), 459  description(aiogram.methods.send_invoice.SendInvoice attribute), 462  description(aiogram.methods.set_chat_description.SetChatDescription CommandScaputTypte), 413  description(aiogram.methods.set_my_description.SetMyDescription Type attribute), 423  description(aiogram.types.bot_command.BotCommand

attribute), 29	attribute), 346
description (aiogram.types.chat.Chat attribute), 37	disable_notification
${\tt description} ({\it aiogram.types.chat\_full\_info.ChatFullInfo}$	$(a iogram. methods. forward\_messages. Forward Messages, forward $
attribute), 59	attribute), 347
description (aiogram.types.game.Game attribute), 301	
attribute), 233	nlineQuery( <b>RixsgltAntincle</b> hods.pin_chat_message.PinChatMessage attribute), 366
${\tt description} ({\it aiogram.types.inline\_query\_result\_cached\_tautout}) \\$	dicsadelet_Indivi QuextRenultCachedDocument
attribute), 239	$(aiogram.methods.send\_animation.Send Animation$
description (aiogram.types.inline_query_result_cached_ attribute), 245	photo.Inlin <b>etOileuts</b> Re&TNoCachedPhoto disable_notification
description(aiogram.types.inline_query_result_cached_	
attribute), 249	attribute), 379
description(aiogram.types.inline_query_result_documen	
attribute), 255	(aiogram.methods.send_contact.SendContact
description(aiogram.types.inline_query_result_photo.In	
attribute), 264	disable_notification
description(aiogram.types.inline_query_result_video.In	
attribute), 268	tribute), 384
description(aiogram.types.input_invoice_message_conta	
attribute), 272	(aiogram.methods.send_document.SendDocument
description (aiogram.types.invoice.Invoice attribute),	attribute), 387
292	disable_notification
DetailedAiogramError, 554	(aiogram.methods.send_game.SendGame
DICE (aiogram.enums.content_type.ContentType at-	attribute), 453
tribute), 473	disable_notification
DICE (aiogram.enums.dice_emoji.DiceEmoji attribute), 478	(aiogram.methods.send_invoice.SendInvoice attribute), 463
DICE (aiogram.types.dice.DiceEmoji attribute), 127	disable_notification
dice(aiogram.types.external_reply_info.ExternalReplyInfo	
attribute), 130	attribute), 390
dice (aiogram.types.message.Message attribute), 160	disable_notification
Dice (class in aiogram.types.dice), 127	(aiogram.methods.send_media_group.SendMediaGroup
DiceEmoji (class in aiogram.enums.dice_emoji), 478	attribute), 392
DiceEmoji (class in aiogram.types.dice), 127	disable_notification
disable() (aiogram.utils.callback_answer.CallbackAnswe	r (aiogram.methods.send_message.SendMessage
method), 579	attribute), 394
disable_content_type_detection	disable_notification
(aiogram.methods.send_document.SendDocumen	t (aiogram.methods.send_photo.SendPhoto
attribute), 387	attribute), 397
disable_content_type_detection	disable_notification
(aiogram.types.input_media_document.InputMed	iaDocume(utiogram.methods.send_poll.SendPoll at-
attribute), 143	tribute), 400
disable_edit_message	disable_notification
(aiogram.methods.set_game_score.SetGameScore	e (aiogram.methods.send_sticker.SendSticker
attribute), 455	attribute), 311
disable_notification	disable_notification
(aiogram.methods.copy_message.CopyMessage	(aiogram.methods.send_venue.SendVenue
attribute), 330	attribute), 402
disable_notification	disable_notification
(aiogram.methods.copy_messages.CopyMessages	(aiogram.methods.send_video.SendVideo
attribute), 332	attribute), 405
disable_notification	disable_notification
$(a iogram. methods. forward\_message. Forward Message and States and States are also as a function of the states are also$	sage (aiogram.methods.send_video_note.SendVideoNote

attribute), 408	attribute), 375
disable_notification	duration (aiogram.methods.send_audio.SendAudio at-
(aiogram.methods.send_voice.SendVoice	tribute), 378
attribute), 410	duration (aiogram.methods.send_video.SendVideo at-
disable_web_page_preview	tribute), 404
(aiogram.methods.edit_message_text.EditMessag attribute), 444	re <b>Cara</b> tion (aiogram.methods.send_video_note.SendVideoNote attribute), 407
disable_web_page_preview	duration (aiogram.methods.send_voice.SendVoice at-
(aiogram.methods.send_message.SendMessage	tribute), 410
attribute), 394	duration (aiogram.types.animation.Animation at-
disable_web_page_preview	tribute), 19
	t <b>RexrEde:son</b> ge <b>diogtant</b> .types.audio.Audio attribute), 19
attribute), 276	${\tt duration} ({\it aiogram.types.input\_media\_animation.Input} {\it MediaAnimation}$
${\tt disabled}$ (a $iogram.utils.callback\_answer.CallbackAnswer.cal$	
property), 579	duration(aiogram.types.input_media_audio.InputMediaAudio
Dispatcher (class in aiogram.dispatcher.dispatcher),	attribute), 142
495	duration(aiogram.types.input_media_video.InputMediaVideo
distance (aiogram.types.proximity_alert_triggered.Proximity_alert_trig	
attribute), 213	duration (aiogram.types.video.Video attribute), 225
DKK (aiogram.enums.currency.Currency attribute), 476 do() (aiogram.types.chat.Chat method), 42	duration(aiogram.types.video_chat_ended.VideoChatEnded attribute), 225
DOCUMENT (aiogram.enums.content_type.ContentType at-	
tribute), 473	duration (aiogram.types.video_note.VideoNote at- tribute), 227
DOCUMENT (aiogram.enums.inline_query_result_type.Inline	<i>"</i>
attribute), 479	DZD (aiogram.enums.currency.Currency attribute), 476
DOCUMENT (aiogram.enums.input_media_type.InputMedia1	
attribute), 480	Ë
document (aiogram.methods.send_document.SendDocume	nedit_caption() (aiogram.types.message.Message
attribute), 386	method) 203
document (aiogram.types.background_type_pattern.Backg	reindTypsPatteThoram.types.message.Message attribute).
attribute), 23	159
<pre>document (aiogram.types.background_type_wallpaper.Bac</pre>	kannindTworWallnane() (aiogram.types.chat.Chat
attribute), 24	method), 41
${\tt document}\ (a iogram. types. external\_reply\_info. ExternalReplace and the property of the$	ledfe_live_location()
attribute), 129	(aiogram.types.message.Message method),
document (aiogram.types.message.Message attribute),	201
159	edit_media() (aiogram.types.message.Message
Document (class in aiogram.types.document), 128	method), 200
document_file_id(aiogram.types.inline_query_result_c	- 1 /- 1 0
<pre>attribute), 239 document_url (aiogram.types.inline_query_result_docum</pre>	(aiogram.types.message.Message method),
attribute), 255	-201-
DOP (aiogram.enums.currency.Currency attribute), 476	edit_text() (aiogram.types.message.Message method),
download() (aiogram.client.bot.Bot method), 486	199
download_file() (aiogram.client.bot.Bot method), 485	
	EditChatInviteLink (class in
DRIVER LICENSE (aiogram enums encrypted passport ele	aiogram.methods.edit_chat_invite_link),
DRIVER_LICENSE (aiogram.enums.encrypted_passport_ele attribute), 478	aiogram.methods.edit_chat_invite_link), ement.EncryptedPassportElement
attribute), 478	aiogram.methods.edit_chat_invite_link), ement.EncryptedPassportElement EDITED_BUSINESS_MESSAGE
	aiogram.methods.edit_chat_invite_link), ement.EncryptedPassportElement EDITED_BUSINESS_MESSAGE (aiogram.enums.update_type.UpdateType
attribute), 478 drop_pending_updates	aiogram.methods.edit_chat_invite_link), ement.EncryptedPassportElement  EDITED_BUSINESS_MESSAGE (aiogram.enums.update_type.UpdateType ok attribute), 484
attribute), 478 drop_pending_updates (aiogram.methods.delete_webhook.DeleteWebhook	aiogram.methods.edit_chat_invite_link), ement.EncryptedPassportElement  EDITED_BUSINESS_MESSAGE
attribute), 478 drop_pending_updates (aiogram.methods.delete_webhook.DeleteWebhoo attribute), 464	aiogram.methods.edit_chat_invite_link), ement.EncryptedPassportElement  EDITED_BUSINESS_MESSAGE (aiogram.enums.update_type.UpdateType ok attribute), 484
attribute), 478 drop_pending_updates	aiogram.methods.edit_chat_invite_link), ement.EncryptedPassportElement  EDITED_BUSINESS_MESSAGE

attribute), 484	$(aiogram.types.chat\_full\_info.ChatFullInfo$
edited_channel_post (aiogram.types.update.Update	attribute), 58
attribute), 298	emoji_status_expiration_date
EDITED_MESSAGE (aiogram.enums.update_type.UpdateTyp	
attribute), 484	emoji_status_expiration_date
edited_message (aiogram.types.update.Update attribute), 298	(aiogram.types.chat_full_info.ChatFullInfo attribute), 58
	EncryptedCredentials (class in
<pre>aiogram.methods.edit_forum_topic), 342</pre>	aiogram.types.encrypted_credentials), 281
- · · · · · · · · · · · · · · · · · · ·	EncryptedPassportElement (class in
<pre>aiogram.methods.edit_general_forum_topic), 343</pre>	aiogram.enums.encrypted_passport_element), 478
	EncryptedPassportElement (class in
aiogram.methods.edit_message_caption), 437	aiogram.types.encrypted_passport_element), 282
EditMessageLiveLocation (class in	<pre>enter() (aiogram.fsm.scene.ScenesManager method),</pre>
$aiogram.methods.edit\_message\_live\_location),$	546
438	<pre>enter() (aiogram.fsm.scene.SceneWizard method), 547</pre>
EditMessageMedia (class in aiogram.methods.edit_message_media), 441	entities (aiogram.methods.edit_message_text.EditMessageText attribute), 444
	entities (aiogram.methods.send_message.SendMessage
aiogram.methods.edit_message_reply_markup),	attribute), 394
442	$entities ({\it aiogram.types.input\_text\_message\_content.InputTextMessageContent.InputCextMessageConten$
EditMessageText (class in	attribute), 276
aiogram.methods.edit_message_text), 443	entities (aiogram.types.message.Message attribute),
EGP (aiogram.enums.currency.Currency attribute), 476	159
element_hash(aiogram.types.passport_element_error_un	
attribute), 291	tribute), 220
	p <b>exdRaxspossEdgev(ai</b> ogram.methods.answer_pre_checkout_query.AnswerPr
attribute), 479	attribute), 456
attribute), 481	v <b>Expre</b> or_message (aiogram.methods.answer_shipping_query.AnswerShippin attribute), 457
<pre>email(aiogram.types.encrypted_passport_element.Encryp</pre>	
attribute), 282	errors (aiogram.methods.set_passport_data_errors.SetPassportDataError
email (aiogram.types.order_info.OrderInfo attribute),	attribute), 470
293	ETB (aiogram.enums.currency.Currency attribute), 476
Email (class in aiogram.utils.formatting), 585	EUR (aiogram.enums.currency.Currency attribute), 476
$\verb"EMOJI" (a iogram.enums.reaction\_type\_type.ReactionTypeTy is a constraint of the property o$	
attribute), 483	event_type (aiogram.types.update.Update property),
emoji (aiogram.methods.send_dice.SendDice attribute),	300
384	exception (aiogram.types.error_event.ErrorEvent at-
emoji (aiogram.methods.send_sticker.SendSticker	tribute), 553
attribute), 311 emoji (aiogram.types.dice.Dice attribute), 127	ExceptionMessageFilter (class in aiogram.filters.exception), 509
	mexiceptions (aiogram.filters.exception.ExceptionTypeFilter
attribute), 215	attribute), 509
emoji (aiogram.types.sticker.Sticker attribute), 279	ExceptionTypeFilter (class in
emoji_list (aiogram.methods.set_sticker_emoji_list.SetS.	• • • • • • • • • • • • • • • • • • • •
attribute), 313	exit() (aiogram.fsm.scene.SceneWizard method), 547
<pre>emoji_list (aiogram.types.input_sticker.InputSticker</pre>	expiration_date(aiogram.types.chat_boost.ChatBoost
attribute), 278	attribute), 52
emoji_status_custom_emoji_id	expire_date(aiogram.methods.create_chat_invite_link.CreateChatInvite
(aiogram.types.chat.Chat attribute), 37	attribute), 333
emoii status custom emoii id	expire date (aiogram methods edit chat invite link EditChatInviteLink

attribute), 341	$\verb file_hash  (a iogram. types. passport\_element\_error\_selfie. PassportElement\_error\_selfie. Pa$
expire_date(aiogram.types.chat_invite_link.ChatInviteLi	
attribute), 61 explanation (aiogram.methods.send_poll.SendPoll at-	file_hash(aiogram.types.passport_element_error_translation_file.Passport_element_eleme
tribute), 399	file_hashes (aiogram.types.passport_element_error_files.PassportEleme
explanation (aiogram.types.poll.Poll attribute), 212	attribute), 286
explanation_entities	file_hashes (aiogram.types.passport_element_error_translation_files.Pa
(aiogram.methods.send_poll.SendPoll at-	attribute), 291
tribute), 399	file_id (aiogram.methods.get_file.GetFile attribute),
explanation_entities (aiogram.types.poll.Poll	354
attribute), 212	file_id (aiogram.types.animation.Animation attribute),
explanation_parse_mode	18
	file_id (aiogram.types.audio.Audio attribute), 19
tribute), 399	file_id (aiogram.types.document.Document attribute),
export() (aiogram.utils.keyboard.InlineKeyboardBuilder	128  File id (in any types fle File attribute) 121
method), 564 export() (aiogram.utils.keyboard.ReplyKeyboardBuilder	file_id (aiogram.types.file.File attribute), 131 file_id (aiogram.types.passport.file_PassportFile_at
method), 565	tribute), 292
method), 42	file_id (aiogram.types.photo_size.PhotoSize attribute), 211
	file_id (aiogram.types.sticker.Sticker attribute), 279
aiogram.methods.export_chat_invite_link),	file_id (aiogram.types.video.Video attribute), 224
344	file_id (aiogram.types.video_note.VideoNote at-
external_reply (aiogram.types.message.Message at-	tribute), 227
tribute), 158 ExternalReplyInfo (class in	file_id (aiogram.types.voice.Voice attribute), 227 file_name (aiogram.types.animation.Animation at-
aiogram.types.external_reply_info), 128	tribute), 19
	file_name (aiogram.types.audio.Audio attribute), 20
556	file_name (aiogram.types.document.Document at-
extract_from() (aiogram.types.message_entity.MessageL	
method), 206	file_name (aiogram.types.video.Video attribute), 225
EYES (aiogram.enums.mask_position_point.MaskPositionPe	
attribute), 480	file_size (aiogram.types.animation.Animation at-
F	tribute), 19
	file_size (aiogram.types.audio.Audio attribute), 20
feed_raw_update() (aiogram.dispatcher.dispatcher.Dispatcher.Dispatcher.dispatcher.dispatcher.Dispatcher.dispatcher.Dispatcher.dispatcher.Dispatcher.dispatcher.dispatcher.dispatcher.Dispatcher.dispat	###@_size (aiogram.types.document.Document at-
method), 495	tribute), 128  file size (diagram times file File attribute) 121
feed_update() (aiogram.dispatcher.dispatcher.Dispatcher.	file_size (aiogram.types.passport_file.PassportFile at-
method), 495 field_name(aiogram.types.passport_element_error_data_	
attribute), 284	file_size (aiogram.types.photo_size.PhotoSize at-
file (aiogram.client.telegram.TelegramAPIServer	tribute), 211
attribute), 14	file_size (aiogram.types.sticker.Sticker attribute), 280
FILE (aiogram.enums.passport_element_error_type.Passpo	
attribute), 482	file_size (aiogram.types.video_note.VideoNote at-
File (class in aiogram.types.file), 131	tribute), 227
file_date (aiogram.types.passport_file.PassportFile at-	file_size (aiogram.types.voice.Voice attribute), 228
tribute), 292	file_unique_id (aiogram.types.animation.Animation
${\tt file\_hash} ({\it aiogram.types.passport\_element\_error\_file.Pa}$	ssportElementib#6)File
attribute), 285	file_unique_id (aiogram.types.audio.Audio attribute),
<pre>file_hash(aiogram.types.passport_element_error_front</pre>	file_unique_id (aiogram.types.document.Document
${\tt file\_hash} (a iogram.types.passport\_element\_error\_reverseleme$	e_side.PassportetementErrorReverseSide
attribute), 288	file_unique_id (aiogram.types.file.File attribute), 131

file_unique_id(aiogram.types.passport_file.PassportFileattribute), 292	leforce_reply (aiogram.types.force_reply.ForceReply at- tribute), 132
file_unique_id (aiogram.types.photo_size.PhotoSize attribute), 211	ForceReply (class in aiogram.types.force_reply), 131  FOREHEAD (aiogram.enums.mask_position_point.MaskPositionPoint
file_unique_id (aiogram.types.sticker.Sticker at-	attribute), 480
tribute), 279	format (aiogram.methods.set_sticker_set_thumbnail.SetStickerSetThumbna
file_unique_id (aiogram.types.video.Video attribute),	attribute), 318
224	format (aiogram.types.input_sticker.InputSticker at-
file_unique_id (aiogram.types.video_note.VideoNote	tribute), 277
attribute), 227	FORUM_TOPIC_CLOSED (aiogram.enums.content_type.ContentType
file_unique_id (aiogram.types.voice.Voice attribute),	attribute), 474
227	<pre>forum_topic_closed (aiogram.types.message.Message</pre>
<pre>file_url() (aiogram.client.telegram.TelegramAPIServer</pre>	attribute), 162
method), 14	FORUM_TOPIC_CREATED
FILES (aiogram.enums.passport_element_error_type.Passp	portElemen <b>(EinrymEnype</b> nums.content_type.ContentType
attribute), 482	attribute), 474
files(aiogram.types.encrypted_passport_element.Encryp	<i>tådRumptopElemene</i> ated
attribute), 282	(aiogram.types.message.Message attribute),
fill(aiogram.types.background_type_fill.BackgroundType	eFill 162
attribute), 23	FORUM_TOPIC_EDITED (aiogram.enums.content_type.ContentType
fill(aiogram.types.background_type_pattern.Background	TypePatter <b>a</b> ttribute), 474
attribute), 23	<pre>forum_topic_edited (aiogram.types.message.Message</pre>
Filter (class in aiogram.filters.base), 510	attribute), 162
filter() (aiogram.filters.callback_data.CallbackData	FORUM_TOPIC_REOPENED
class method), 507	(aiogram.enums.content_type.ContentType
FIND_LOCATION (aiogram.enums.chat_action.ChatAction	attribute), 474
attribute), 472	forum_topic_reopened
find_location() (aiogram.utils.chat_action.ChatActionS	
class method), 571	162
${\tt first\_name} \ (a iogram.methods.send\_contact.SendContact$	
attribute), 382	ForumTopicClosed (class in
first_name (aiogram.types.chat.Chat attribute), 35	aiogram.types.forum_topic_closed), 132
first_name (aiogram.types.chat_full_info.ChatFullInfo	
attribute), 58	aiogram.types.forum_topic_created), 133
first_name (aiogram.types.contact.Contact attribute),	
127	aiogram.types.forum_topic_edited), 133
first_name(aiogram.types.inline_query_result_contact.In	
attribute), 252	aiogram.types.forum_topic_reopened), 134
first_name(aiogram.types.input_contact_message_conte	
attribute), 271	200
first_name (aiogram.types.shared_user.SharedUser at-	forward_date (aiogram.types.message.Message at-
tribute), 218	tribute), 163
first_name (aiogram.types.user.User attribute), 221	forward_from (aiogram.types.message.Message at-
first_name (aiogram.utils.web_app.WebAppUser	tribute), 163
attribute), 575	forward_from_chat (aiogram.types.message.Message
FOOTBALL (aiogram.enums.dice_emoji.DiceEmoji	attribute), 163
attribute), 478	forward_from_message_id
FOOTBALL (aiogram.types.dice.DiceEmoji attribute), 128	(aiogram.types.message.Message attribute),
for_channels (aiogram.methods.get_my_default_adminis	• • •
<pre>attribute), 358 for_channels (aiogram.methods.set_my_default_adminis.</pre>	forward_origin (aiogram.types.message.Message at-
attribute), 423	trator_rign <b>usione</b> jylægautAaministratorRignis forward_sender_name
force (aiogram.methods.set_game_score.SetGameScore	(aiogram.types.message.Message attribute),
attribute), 455	(diogram.types.message.message airrome), 163
	100

```
forward_signature (aiogram.types.message.Message
                                                                                                                                                                                                                                                                                          attribute), 106
                                        attribute), 163
                                                                                                                                                                                                                                                  from_user(aiogram.types.chosen inline result.ChosenInlineResult
forward_text
                                                                                                                                                                                                                                                                                          attribute), 229
                                                                                             (aiogram.types.login url.LoginUrl
                                                                                                                                                                                                                                                  from_user (aiogram.types.inline_query.InlineQuery at-
                                        attribute), 153
ForwardMessage
                                                                                                                                         (class
                                                                                                                                                                                                                               in
                                                                                                                                                                                                                                                                                          tribute), 230
                                        aiogram.methods.forward message), 345
                                                                                                                                                                                                                                                  from_user (aiogram.types.message.Message attribute),
ForwardMessages
                                                                                                                                            (class
                                                                                                                                                                                                                               in
                                        aiogram.methods.forward_messages), 347
                                                                                                                                                                                                                                                  from_user(aiogram.types.pre_checkout_query.PreCheckoutQuery
foursquare_id(aiogram.methods.send venue.SendVenue
                                                                                                                                                                                                                                                                                           attribute), 294
                                        attribute), 402
                                                                                                                                                                                                                                                  from_user(aiogram.types.shipping_query.ShippingQuery
foursquare_id(aiogram.types.inline_query_result_venue.InlineQuearttRibruthY,earth
                                                                                                                                                                                                                                                 {\tt FRONT\_SIDE}\,(aiogram.enums.passport\_element\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement\_error\_type.PassportElement
                                        attribute), 266
foursquare_id(aiogram.types.input_venue_message_content.InputVatmibMe)s4%2Content
                                        attribute), 277
                                                                                                                                                                                                                                                  front_side(aiogram.types.encrypted_passport_element.EncryptedPassport_element.
foursquare_id (aiogram.types.venue.Venue attribute),
                                                                                                                                                                                                                                                                                          attribute), 282
                                                                                                                                                                                                                                                  FSInputFile (class in aiogram.types.input_file), 139,
foursquare_type (aiogram.methods.send_venue.SendVenue
                                                                                                                                                                                                                                                                                          487
                                        attribute), 402
                                                                                                                                                                                                                                                 FSMI18nMiddleware
                                                                                                                                                                                                                                                                                                                                                                                                   (class
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 in
foursquare_type (aiogram.types.inline_query_result_venue.InlineQuiogyRosulttVentl&n.middleware), 568
                                                                                                                                                                                                                                                  full_name (aiogram.types.chat.Chat property), 39
                                        attribute), 266
foursquare_type (aiogram.types.input_venue_message_condent_theme VaiogeMessygecCoostentser property), 221
                                        attribute), 277
                                                                                                                                                                                                                                                 G
                                                                                                                                                                                                                            at-
foursquare_type
                                                                                             (aiogram.types.venue.Venue
                                        tribute), 224
                                                                                                                                                                                                                                                  GAME
                                                                                                                                                                                                                                                                                     (aiogram.enums.content_type.ContentType
from_attachment_menu
                                                                                                                                                                                                                                                                                          tribute), 473
                                        (aiogram.types.write\_access\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAccess\_allowed.WriteAcc
                                        attribute), 229
                                                                                                                                                                                                                                                                                          attribute), 479
{\tt from\_base()}\ (a iogram. client. telegram. Telegram API Server {\tt game}\ (a iogram. types. external\_reply\_info. ExternalReplyInfo)
                                        class method), 14
                                                                                                                                                                                                                                                                                          attribute), 130
{\tt from\_chat\_id} \ (a iogram. methods. copy\_message. CopyMess {\tt game} \ (a iogram. types. message. Message \ attribute), \ 160 \\
                                        attribute), 329
                                                                                                                                                                                                                                                  Game (class in aiogram.types.game), 301
\verb|from_chat_id|| (a iogram.methods.copy_messages.CopyMessages\_short_name|| (a iogram.methods.send_game.SendGame)|| (a iogram.methods.copy_messages.CopyMessages)|| (a iogram.methods.copy_messages)|| (a iogram.methods.copy_mes
                                        attribute), 331
                                                                                                                                                                                                                                                                                           attribute), 453
{\tt from\_chat\_id} \ (a iogram. methods. forward\_message. Forward\_message.
                                         attribute), 345
                                                                                                                                                                                                                                                                                          attribute), 34
\label{lem:chat_id} \textbf{from\_chat\_id} (\textit{aiogram.methods.forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.Forward\_messages.
                                        attribute), 347
                                                                                                                                                                                                                                                                                           attribute), 256
{\tt from\_file()} \ ({\it aiogram.types.input\_file.BufferedInputFile} \ \ {\tt GameHighScore}
                                                                                                                                                                                                                                                                                                                                                                                        (class
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 in
                                         class method), 139
                                                                                                                                                                                                                                                                                          aiogram.types.game_high_score), 302
from_markup() (aiogram.utils.keyboard.InlineKeyboardBuckler(aiogram.enums.currency.Currency attribute), 476
                                        class method), 564
                                                                                                                                                                                                                                                  GEL (aiogram.enums.currency.Currency attribute), 476
{\tt from\_markup()} \ (a iogram.utils. keyboard. Reply Keyboard Builtoner AL\_FORUM\_TOPIC\_HIDDEN
                                        class method), 565
                                                                                                                                                                                                                                                                                          (aiogram.enums.content_type.ContentType
from_request (aiogram.types.write_access_allowed.WriteAccessAllowed), 474
                                        attribute), 229
                                                                                                                                                                                                                                                  general_forum_topic_hidden
                                                                      (aiogram.fsm.storage.redis.RedisStorage
from_url()
                                                                                                                                                                                                                                                                                           (aiogram.types.message.Message
                                                                                                                                                                                                                                                                                                                                                                                                                                                 attribute),
                                        class method), 529
{\tt from\_user} (a iogram. handlers. callback\_query. CallbackQuery. CallbackQuery
                                        property), 558
                                                                                                                                                                                                                                                                                           (aiogram.enums.content_type.ContentType
 from_user(aiogram.types.callback_query.CallbackQuery
                                                                                                                                                                                                                                                                                           attribute), 474
                                        attribute), 33
                                                                                                                                                                                                                                                  general_forum_topic_unhidden
from_user(aiogram.types.chat_join_request.ChatJoinRequest
                                                                                                                                                                                                                                                                                           (aiogram.types.message.Message
                                                                                                                                                                                                                                                                                                                                                                                                                                                 attribute),
                                        attribute), 61
from_user(aiogram.types.chat member updated.ChatMemberUpdated
```

GeneralForumTopicHidden (class in	<pre>aiogram.methods.get_my_default_administrator_rights), 358</pre>
aiogram.types.general_forum_topic_hidden), 134	GetMyDescription (class in
GeneralForumTopicUnhidden (class in	aiogram.methods.get_my_description), 359
aiogram.types.general_forum_topic_unhidden),	GetMyName (class in aiogram.methods.get_my_name),
134	360)
get() (aiogram.fsm.scene.SceneRegistry method), 546	GetMyShortDescription (class in
get_administrators() (aiogram.types.chat.Chat method), 40	aiogram.methods.get_my_short_description), 361
get_data() (aiogram.fsm.scene.SceneWizard method), 548	GetStickerSet (class in aiogram.methods.get_sticker_set), 308
get_data() (aiogram.fsm.storage.base.BaseStorage	
method), 531	465
<pre>get_flag() (in module aiogram.dispatcher.flags), 556</pre>	GetUserChatBoosts (class in
<pre>get_locale() (aiogram.utils.i18n.middleware.I18nMiddl</pre>	eware aiogram.methods.get_user_chat_boosts), 361
<pre>get_member() (aiogram.types.chat.Chat method), 44</pre>	GetUserProfilePhotos (class in
<pre>get_member_count()</pre>	aiogram.methods.get_user_profile_photos), 362
<pre>get_profile_photos() (aiogram.types.user.User</pre>	GetWebhookInfo (class in
method), 221	aiogram.methods.get_webhook_info), 467
<pre>get_state() (aiogram.fsm.storage.base.BaseStorage</pre>	GIF (aiogram.enums.inline_query_result_type.InlineQueryResultType
method), 530	attribute), 479
<pre>get_url() (aiogram.types.message.Message method),</pre>	<pre>gif_duration(aiogram.types.inline_query_result_gif.InlineQueryResult attribute), 258</pre>
GetBusinessConnection (class in	$\verb"gif_file_id" (a iogram.types.in line_query_result_cached_gif.In lineQuery_result_cached_gif.In lineQuery_result_cached_g$
aiogram.methods.get_business_connection),	attribute), 240
348 GetChat (class in aiogram.methods.get_chat), 349	<pre>gif_height (aiogram.types.inline_query_result_gif.InlineQueryResultGiter) attribute), 257</pre>
GetChatAdministrators (class in	$\verb"gif\_url" (a iogram.types.inline\_query\_result\_gif.InlineQueryResultGif")$
$aiogram.methods.get\_chat\_administrators),$	attribute), 257
350	$\verb"gif_width" (a iogram.types.inline\_query\_result\_gif.InlineQueryResultGif") and the property of the property$
GetChatMember (class in	attribute), 257
aiogram.methods.get_chat_member), 351	GIFT_CODE (aiogram.enums.chat_boost_source_type.ChatBoostSourceType.
GetChatMemberCount (class in	attribute), 472
aiogram.methods.get_chat_member_count), 352	GIVEAWAY (aiogram.enums.chat_boost_source_type.ChatBoostSourceType attribute), 472
	GIVEAWAY (aiogram.enums.content_type.ContentType at-
aiogram.methods.get_chat_menu_button),	tribute), 474
353	giveaway (aiogram.types.external_reply_info.ExternalReplyInfo
GetCustomEmojiStickers (class in	attribute), 130
aiogram.methods.get_custom_emoji_stickers),	giveaway (aiogram.types.message.Message attribute),
307	162
<pre>GetFile (class in aiogram.methods.get_file), 354</pre>	Giveaway (class in aiogram.types.giveaway), 134
	GIVEAWAY_COMPLETED (aiogram.enums.content_type.ContentType
aiogram.methods.get_forum_topic_icon_stickers	
355	giveaway_completed (aiogram.types.message.Message
GetGameHighScores (class in	attribute), 162
aiogram.methods.get_game_high_scores), 451	GIVEAWAY_CREATED (aiogram.enums.content_type.ContentType
GetMe (class in aiogram.methods.get_me), 356	attribute), 474 giveaway_created (aiogram.types.message.Message
GetMyCommands (class in diogram.memous.get_me), 550	attribute), 162
aiogram.methods.get_my_commands), 357	${\tt giveaway\_message} \ (a iogram. types. giveaway\_completed. GiveawayCompleted. Giveawa$
GetMvDefaultAdministratorRights (class in	attribute) 135

giveaway_message_id	has_custom_certificate
	atBoostSout <b>ceiGjvæmstyp</b> es.webhook_info.WebhookInfo
attribute), 55	attribute), 300
giveaway_message_id	has_hidden_members (aiogram.types.chat.Chat at-
(aiogram.types.giveaway_winners.GiveawayWin	. 0 11
attribute), 136	has_hidden_members(aiogram.types.chat_full_info.ChatFullInfo
GIVEAWAY_WINNERS (aiogram.enums.content_type.Content	
attribute), 474	has_media_spoiler(aiogram.types.external_reply_info.ExternalReplyInj
giveaway_winners(aiogram.types.external_reply_info.E	
attribute), 130	has_media_spoiler (aiogram.types.message.Message
<pre>giveaway_winners (aiogram.types.message.Message</pre>	attribute), 160
attribute), 162	has_private_forwards (aiogram.types.chat.Chat at-
GiveawayCompleted (class in	tribute), 37
aiogram.types.giveaway_completed), 135	has_private_forwards
GiveawayCreated (class in	(aiogram.types.chat_full_info.ChatFullInfo
aiogram.types.giveaway_created), 136	attribute), 59
GiveawayWinners (class in	has_protected_content (aiogram.types.chat.Chat at-
aiogram.types.giveaway_winners), 136	tribute), 37
<pre>google_place_id(aiogram.methods.send_venue.SendVenu</pre>	ndmas_protected_content
attribute), 402	(aiogram.types.chat_full_info.ChatFullInfo
<pre>google_place_id(aiogram.types.inline_query_result_ve</pre>	nue.InlineQ <b>uteriyRies)</b> ul <b>5</b> Venue
attribute), 266	has_protected_content
<pre>google_place_id(aiogram.types.input_venue_message_</pre>	content.Inp <b>(aWegnunMegsergnConaega</b> tMessage attribute),
attribute), 277	159
<pre>google_place_id (aiogram.types.venue.Venue at-</pre>	has_public_winners(aiogram.types.giveaway.Giveaway
tribute), 224	attribute), 135
<pre>google_place_type (aiogram.methods.send_venue.Send</pre>	
attribute), 402	(aiogram.types.chat.Chat attribute), 37
<pre>google_place_type (aiogram.types.inline_query_result_</pre>	
attribute), 266	(aiogram.types.chat_full_info.ChatFullInfo
<pre>google_place_type (aiogram.types.input_venue_message)</pre>	
attribute), 277	$\verb has_spoiler  (a iogram.methods.send_animation.SendAnimation $
google_place_type (aiogram.types.venue.Venue	attribute), 376
attribute), 224	has_spoiler (aiogram.methods.send_photo.SendPhoto
goto() (aiogram.fsm.scene.SceneWizard method), 548	attribute), 396
GREEN (aiogram.enums.topic_icon_color.TopicIconColor	
attribute), 484	attribute), 405
•	has_spoiler(aiogram.types.input_media_animation.InputMediaAnimatic
473	attribute), 141
	tehaBypspoiler(aiogram.types.input_media_photo.InputMediaPhoto
attribute), 474	attribute), 144
<pre>group_chat_created (aiogram.types.message.Message</pre>	has_spoiler(aiogram.types.input_media_video.InputMediaVideo
attribute), 160	attribute), 146
GTQ (aiogram.enums.currency.Currency attribute), 476	has_visible_history (aiogram.types.chat.Chat
Н	attribute), 37
П	has_visible_history
handlers (aiogram.fsm.scene.SceneConfig attribute), 547	(aiogram.types.chat_full_info.ChatFullInfo attribute), 59
has_aggressive_anti_spam_enabled	$hash ({\it aiogram.types.encrypted\_credentials.} Encrypted Credentials$
(aiogram.types.chat.Chat attribute), 37	attribute), 281
has_aggressive_anti_spam_enabled	$hash (a iogram. types. encrypted\_passport\_element. Encrypted Passport Element (a iogram. types. encrypted\_passport\_element) and the properties of the prop$
(aiogram.types.chat_full_info.ChatFullInfo	attribute), 282
attribute), 59	hash (aiogram.utils.web_app.WebAppInitData attribute),
	575

```
HASHTAG (aiogram.enums.message entity type.MessageEnthythhpetext (aiogram.types.message.Message property),
                           attribute), 481
HashTag (class in aiogram.utils.formatting), 584
                                                                                                                                                               HUF (aiogram.enums.currency.Currency attribute), 476
heading(aiogram.methods.edit\_message\_live\_location.EditMessageLiveLocation)
                           attribute), 440
heading (aiogram.methods.send location.SendLocation
                                                                                                                                                               I18nMiddleware
                                                                                                                                                                                                                                                          (class
                                                                                                                                                                                                                                                                                                                  in
                          attribute), 389
                                                                                                                                                                                          aiogram.utils.i18n.middleware), 568
\textbf{heading} (a iogram. types. in line\_query\_result\_location. In line \underline{Query\_Result} (a tog from .methods. create\_forum\_topic. CreateForumTopic .methods. create\_forum\_topic .me
                           attribute), 260
                                                                                                                                                                                          attribute), 334
heading (aiogram.types.input_location_message_content.InputLocationMessageContent.types.forum_topic.ForumTopic at-
                          attribute), 275
                                                                                                                                                                                          tribute), 132
heading (aiogram.types.location.Location attribute),
                                                                                                                                                               \verb"icon_color" (a iogram.types.forum\_topic\_created.ForumTopicCreated")
                                                                                                                                                                                          attribute), 133
\verb|height| (a iogram.methods.send\_animation.SendAnimation | \verb|icon\_custom\_emoji\_id|)|
                          attribute), 375
                                                                                                                                                                                          (aiogram.methods.create forum topic.CreateForumTopic
height
                         (aiogram.methods.send_video.SendVideo
                                                                                                                                                                                          attribute), 334
                           tribute), 404
                                                                                                                                                               icon_custom_emoji_id
height (aiogram.types.animation.Animation attribute),
                                                                                                                                                                                          (aiogram.methods.edit_forum_topic.EditForumTopic
                                                                                                                                                                                          attribute), 342
height (aiogram.types.input_media_animation.InputMediaArimatiagtom_emoji_id
                          attribute), 141
                                                                                                                                                                                          (aiogram.types.forum_topic.ForumTopic
height(aiogram.types.input\_media\_video.InputMediaVideo)
                                                                                                                                                                                          attribute), 132
                          attribute), 145
                                                                                                                                                               icon_custom_emoji_id
height (aiogram.types.photo size.PhotoSize attribute),
                                                                                                                                                                                          (aiogram.types.forum_topic_created.ForumTopicCreated
                           211
                                                                                                                                                                                          attribute), 133
height (aiogram.types.sticker.Sticker attribute), 279
                                                                                                                                                               icon_custom_emoji_id
height (aiogram.types.video.Video attribute), 225
                                                                                                                                                                                          (aiogram.types.forum topic edited.ForumTopicEdited
HIDDEN_USER (aiogram.enums.message_origin_type.MessageOriginTyperibute), 133
                          attribute), 482
                                                                                                                                                               id(aiogram.types.business connection.BusinessConnection
hide_url (aiogram.types.inline_query_result_article.InlineQueryResultApticle, 30
                          attribute), 233
                                                                                                                                                               id (aiogram.types.callback_query.CallbackQuery at-
HideGeneralForumTopic
                                                                                                       (class
                                                                                                                                                   in
                                                                                                                                                                                          tribute), 33
                          aiogram.methods.hide_general_forum_topic),
                                                                                                                                                               id (aiogram.types.chat.Chat attribute), 35
                                                                                                                                                               id (aiogram.types.chat_full_info.ChatFullInfo attribute),
HKD (aiogram.enums.currency.Currency attribute), 476
HNL (aiogram.enums.currency.Currency attribute), 476
                                                                                                                                                               id (aiogram.types.inline_query.InlineQuery attribute),
horizontal_accuracy
                                                                                                                                                                                          230
                          (aiogram.methods.edit\_message\_live\_location. Edit \texttt{MessageLive} \\ \texttt{Joesation} \\ \texttt{ne\_query\_result\_article}. In line Query \\ \texttt{ResultArticle} \\ \texttt{ne\_query\_result\_article}. \\ \texttt{Ne\_quer
                          attribute), 439
                                                                                                                                                                                          attribute), 233
horizontal_accuracy
                                                                                                                                                               id(aiogram.types.inline_query_result_audio.InlineQueryResultAudio
                           (aiogram.methods.send location.SendLocation
                                                                                                                                                                                          attribute), 234
                          attribute), 389
                                                                                                                                                               id(aiogram.types.inline_query_result_cached_audio.InlineQueryResultCa
horizontal_accuracy
                                                                                                                                                                                          attribute), 236
                          (aiogram.types.inline\_query\_result\_location.Inline QueryResult \cite{Apestina} and the permitting and the 
                          attribute), 260
                                                                                                                                                                                          attribute), 239
horizontal_accuracy
                                                                                                                                                               id(aiogram.types.inline_query_result_cached_gif.InlineQueryResultCache
                           (aiogram.types.input_location_message_content.InputLocation_Hessage_Content
                          attribute), 274
                                                                                                                                                               id(aiogram.types.inline_query_result_cached_mpeg4_gif.InlineQueryResu
horizontal_accuracy
                                                                                                                                                                                          attribute), 243
                           (aiogram.types.location.Location
                                                                                                                              attribute),
                                                                                                                                                               id(aiogram.types.inline query result cached photo.InlineQueryResultCa
                                                                                                                                                                                          attribute), 245
HRK (aiogram.enums.currency.Currency attribute), 476
                                                                                                                                                               id(aiogram.types.inline_query_result_cached_sticker.InlineQueryResultCa
HTML (aiogram.enums.parse mode.ParseMode attribute),
```

482

attribute), 247

```
id (aiogram.types.inline_query_result_cached_video.InlineQueryResult@ibahed,Video
         attribute), 249
                                                       inline_message_id(aiogram.methods.edit message text.EditMessageTe.
id (aiogram.types.inline_query_result_cached_voice.InlineQueryResult@ibulued).Volce
                                                       inline_message_id(aiogram.methods.get_game_high_scores.GetGameH
         attribute), 251
id (aiogram.types.inline_query_result_contact.InlineQueryResultContatttibute), 452
         attribute), 252
                                                       inline_message_id(aiogram.methods.set game score.SetGameScore
id (aiogram.types.inline query result document.InlineQueryResultDatwibente), 455
                                                       inline_message_id(aiogram.methods.stop_message_live_location.Stop)
         attribute), 254
id (aiogram.types.inline_query_result_game.InlineQueryResultGame attribute), 446
                                                       inline_message_id(aiogram.types.callback_query.CallbackQuery
         attribute), 256
id(aiogram.types.inline_query_result_gif.InlineQueryResultGif
                                                                 attribute), 34
         attribute), 257
                                                       inline_message_id(aiogram.types.chosen_inline_result.ChosenInlineRe
id (aiogram.types.inline_query_result_location.InlineQueryResultLocattioihute), 230
         attribute), 259
                                                       inline_message_id(aiogram.types.sent_web_app_message.SentWebApp
id (aiogram.types.inline_query_result_mpeg4_gif.InlineQueryResultMiparglandie), 277
         attribute), 262
                                                       INLINE_QUERY (aiogram.enums.update_type.UpdateType
id (aiogram.types.inline_query_result_photo.InlineQueryResultPhotoattribute), 484
         attribute), 263
                                                       inline_query (aiogram.types.update.Update attribute),
id(aiogram.types.inline_query_result_venue.InlineQueryResultVenue299
                                                       inline_query_id(aiogram.methods.answer_inline_query.AnswerInlineQ
         attribute), 265
id (aiogram.types.inline_query_result_video.InlineQueryResultVideo attribute), 448
         attribute), 267
                                                       InlineKeyboardBuilder
                                                                                           (class
                                                                                                           in
id (aiogram.types.inline_query_result_voice.InlineQueryResultVoice aiogram.utils.keyboard), 563
         attribute), 269
                                                       InlineKeyboardButton
                                                                                           (class
                                                                                                           in
id (aiogram.types.poll.Poll attribute), 211
                                                                 aiogram.types.inline_keyboard_button), 137
id(aiogram.types.pre_checkout_query.PreCheckoutQuery InlineKeyboardMarkup)
                                                                                          (class
                                                                                                           in
         attribute), 294
                                                                 aiogram.types.inline_keyboard_markup),
id (aiogram.types.shipping_option.ShippingOption at-
         tribute), 295
                                                       InlineQuery (class in aiogram.types.inline_query), 230
    (aiogram.types.shipping_query.ShippingQuery
                                                       InlineQueryResult
                                                                                        (class
                                                                                                           in
         tribute), 296
                                                                 aiogram.types.inline_query_result), 231
id (aiogram.types.story.Story attribute), 219
                                                       InlineQueryResultArticle
                                                                                             (class
                                                                                                           in
id (aiogram.types.user.User attribute), 221
                                                                 aiogram.types.inline\_query\_result\_article),
id (aiogram.utils.web_app.WebAppChat attribute), 576
                                                                 232
id (aiogram.utils.web app.WebAppUser attribute), 575
                                                       InlineQueryResultAudio
                                                                                                           in
IDENTITY_CARD (aiogram.enums.encrypted_passport_element.EncrypticdPassprope&lintievet_query_result_audio),
         attribute), 478
                                                                 233
IDR (aiogram.enums.currency.Currency attribute), 476
                                                       InlineQueryResultCachedAudio
                                                                                               (class
                                                                                                           in
ILS (aiogram.enums.currency.Currency attribute), 476
                                                                 aiogram.types.inline_query_result_cached_audio),
                                                                 235
InaccessibleMessage
                                  (class
         aiogram.types.inaccessible message), 137
                                                       InlineQueryResultCachedDocument
                                                                                                 (class
                                                                                                           in
include_router() (aiogram.dispatcher.router.Router
                                                                 aiogram.types.inline_query_result_cached_document),
         method), 489
include_routers() (aiogram.dispatcher.router.Router
                                                       InlineQueryResultCachedGif
                                                                                              (class
                                                                                                           in
         method), 489
                                                                 aiogram.types.inline_query_result_cached_gif),
inline_keyboard(aiogram.types.inline_keyboard_markup.InlineKeyB@ardMarkup
                                                       InlineQueryResultCachedMpeg4Gif
         attribute), 139
                                                                                                 (class
inline_message_id (aiogram.methods.edit_message_caption.EditMessage@appiexinline_query_result_cached_mpeg4_gif),
         attribute), 437
inline_message_id (aiogram.methods.edit_message_live InlinineOufadiyHessulgeLiveHedPhiorbo
                                                                                               (class
                                                                                                           in
         attribute), 439
                                                                 aiogram.types.inline_query_result_cached_photo),
inline_message_id(aiogram.methods.edit_message_media.EditMe\sageMedia
         attribute), 441
                                                       InlineQueryResultCachedSticker
                                                                                                 (class
                                                                                                           in
inline_message_id(aiogram.methods.edit message reply markupdidioMnessyngeRinfluMankeny result cached sticker),
```

245	input_message_content
InlineQueryResultCachedVideo (class in	
aiogram.types.inline_query_result_cached_vid	eo), attribute), 237
247	input_message_content
InlineQueryResultCachedVoice (class in	· · · · · · · · · · · · · · · · · · ·
aiogram.types.inline_query_result_cached_void	
250	input_message_content
InlineQueryResultContact (class in	
aiogram.types.inline_query_result_contact), 252	<pre>attribute), 241 input_message_content</pre>
InlineQueryResultDocument (class in	
aiogram.types.inline_query_result_document),	
253	input_message_content
InlineQueryResultGame (class in	
<pre>aiogram.types.inline_query_result_game),</pre>	attribute), 245
255	input_message_content
InlineQueryResultGif (class in	
<pre>aiogram.types.inline_query_result_gif), 256</pre>	attribute), 247
•	input_message_content
aiogram.types.inline_query_result_location), 258	(aiogram.types.inline_query_result_cached_video.InlineQueryReattribute), 249
	n_input_message_content
aiogram.types.inline_query_result_mpeg4_gif) 260	), (aiogram.types.inline_query_result_cached_voice.InlineQueryRe attribute), 251
InlineQueryResultPhoto (class in	input_message_content
<pre>aiogram.types.inline_query_result_photo), 263</pre>	(aiogram.types.inline_query_result_contact.InlineQueryResultCoattribute), 253
InlineQueryResultsButton (class in	n input_message_content
aiogram.types.inline_query_results_button), 270	(aiogram.types.inline_query_result_document.InlineQueryResult attribute), 255
InlineQueryResultType (class in	n input_message_content
<pre>aiogram.enums.inline_query_result_type), 479</pre>	(aiogram.types.inline_query_result_gif.InlineQueryResultGif attribute), 258
InlineQueryResultVenue (class in	input_message_content
<pre>aiogram.types.inline_query_result_venue), 265</pre>	(aiogram.types.inline_query_result_location.InlineQueryResultLattribute), 260
	n input_message_content
aiogram.types.inline_query_result_video), 266	(aiogram.types.inline_query_result_mpeg4_gif.InlineQueryResulatribute), 262
InlineQueryResultVoice (class in	1 - 5 -
aiogram.types.inline_query_result_voice), 268	(aiogram.types.inline_query_result_photo.InlineQueryResultPhoattribute), 264
<pre>input_field_placeholder</pre>	input_message_content
(aiogram.types.force_reply.ForceReply at- tribute), 132	<ul> <li>(aiogram.types.inline_query_result_venue.InlineQueryResultVenue)</li> <li>attribute)</li> <li>266</li> </ul>
<pre>input_field_placeholder</pre>	input_message_content
(aiogram.types.reply_keyboard_markup.Replylattribute), 216	KeyboardMar <b>kaip</b> gram.types.inline_query_result_video.InlineQueryResultVide attribute), 268
input_message_content	input_message_content
	neQueryResu <b>ltAngirthe</b> .types.inline_query_result_voice.InlineQueryResultVoic attribute), 270
input_message_content	InputContactMessageContent (class in
(aiogram.types.inline_query_result_audio.Inlinattribute), 235	neQueryResu <b>ltiAuglio</b> m.types.input_contact_message_content), 271

<pre>InputFile (class in aiogram.types.input_file), 139 InputInvoiceMessageContent (class in</pre>	invoice (aiogram.types.message.Message attribute), 161 Invoice (class in aiogram.types.invoice), 292
<pre>aiogram.types.input_invoice_message_content), 271</pre>	<pre>invoice_payload(aiogram.types.pre_checkout_query.PreCheckoutQuer attribute), 294</pre>
InputLocationMessageContent (class in	<pre>invoice_payload(aiogram.types.shipping_query.ShippingQuery</pre>
aiogram.types.input_location_message_content),	
274	invoice_payload(aiogram.types.successful_payment.SuccessfulPayment
<pre>InputMedia (class in aiogram.types.input_media), 140</pre>	attribute), 297
InputMediaAnimation (class in	<pre>ip_address(aiogram.methods.set_webhook.SetWebhook</pre>
aiogram.types.input_media_animation), 140	attribute), 468
InputMediaAudio (class in	<pre>ip_address (aiogram.types.webhook_info.WebhookInfo</pre>
aiogram.types.input_media_audio), 141	attribute), 300
InputMediaDocument (class in	<pre>ip_filter_middleware()</pre>
aiogram.types.input_media_document), 142	aiogram.webhook.aiohttp_server), 516
InputMediaPhoto (class in	IPFilter (class in aiogram.webhook.security), 516
aiogram.types.input_media_photo), 144	is_animated (aiogram.types.sticker.Sticker attribute),
InputMediaType (class in	279
aiogram.enums.input_media_type), 480	is_animated (aiogram.types.sticker_set.StickerSet at-
InputMediaVideo (class in	tribute), 281
aiogram.types.input_media_video), 144	is_anonymous (aiogram.methods.promote_chat_member.PromoteChatMe
InputMessageContent (class in	attribute), 368
aiogram.types.input_message_content), 275	is_anonymous (aiogram.methods.send_poll.SendPoll at-
InputPollOption (class in	tribute), 399
aiogram.types.input_poll_option), 146	is_anonymous (aiogram.types.chat_administrator_rights.ChatAdministrator
InputSticker (class in aiogram.types.input_sticker),	attribute), 50
277	is_anonymous (aiogram.types.chat_member_administrator.ChatMember.
InputTextMessageContent (class in	attribute), 100
	is anonymous (alogram types chat member owner ('hatMember()wner
aiogram.types.input_text_message_content),	is_anonymous (aiogram.types.chat_member_owner.ChatMemberOwner attribute) 103
275	attribute), 103
275 InputVenueMessageContent (class in	attribute), 103 is_anonymous (aiogram.types.poll.Poll attribute), 211
275 InputVenueMessageContent (class in aiogram.types.input_venue_message_content),	attribute), 103 is_anonymous (aiogram.types.poll.Poll attribute), 211 is_automatic_forward
275 InputVenueMessageContent (class in aiogram.types.input_venue_message_content), 276	attribute), 103 is_anonymous (aiogram.types.poll.Poll attribute), 211 is_automatic_forward (aiogram.types.message.Message attribute),
275 InputVenueMessageContent (class in aiogram.types.input_venue_message_content), 276 INR (aiogram.enums.currency.Currency attribute), 476	attribute), 103 is_anonymous (aiogram.types.poll.Poll attribute), 211 is_automatic_forward
275 InputVenueMessageContent (class in aiogram.types.input_venue_message_content), 276 INR (aiogram.enums.currency.Currency attribute), 476 intensity (aiogram.types.background_type_pattern.Backgroun	attribute), 103 is_anonymous (aiogram.types.poll.Poll attribute), 211 is_automatic_forward
275 InputVenueMessageContent (class in aiogram.types.input_venue_message_content), 276 INR (aiogram.enums.currency.Currency attribute), 476 intensity (aiogram.types.background_type_pattern.Background_type), 24	attribute), 103 is_anonymous (aiogram.types.poll.Poll attribute), 211 is_automatic_forward
InputVenueMessageContent (class in aiogram.types.input_venue_message_content), 276  INR (aiogram.enums.currency.Currency attribute), 476 intensity (aiogram.types.background_type_pattern.Background_type), 24  INTERNAL_PASSPORT (aiogram.enums.encrypted_passport)	attribute), 103 is_anonymous (aiogram.types.poll.Poll attribute), 211 is_automatic_forward
InputVenueMessageContent (class in aiogram.types.input_venue_message_content), 276  INR (aiogram.enums.currency.Currency attribute), 476  intensity (aiogram.types.background_type_pattern.Background_	attribute), 103 is_anonymous (aiogram.types.poll.Poll attribute), 211 is_automatic_forward
InputVenueMessageContent (class in aiogram.types.input_venue_message_content), 276  INR (aiogram.enums.currency.Currency attribute), 476 intensity (aiogram.types.background_type_pattern.Background_t	attribute), 103 is_anonymous (aiogram.types.poll.Poll attribute), 211 is_automatic_forward
InputVenueMessageContent (class in aiogram.types.input_venue_message_content), 276  INR (aiogram.enums.currency.Currency attribute), 476 intensity (aiogram.types.background_type_pattern.Backgattribute), 24  INTERNAL_PASSPORT (aiogram.enums.encrypted_passport attribute), 478 invite_link (aiogram.methods.edit_chat_invite_link.Edit attribute), 341	attribute), 103  is_anonymous (aiogram.types.poll.Poll attribute), 211  is_automatic_forward
InputVenueMessageContent (class in aiogram.types.input_venue_message_content), 276  INR (aiogram.enums.currency.Currency attribute), 476 intensity (aiogram.types.background_type_pattern.Backgattribute), 24  INTERNAL_PASSPORT (aiogram.enums.encrypted_passport attribute), 478 invite_link (aiogram.methods.edit_chat_invite_link.Edit attribute), 341 invite_link (aiogram.methods.revoke_chat_invite_link.Edit attribute).	attribute), 103  is_anonymous (aiogram.types.poll.Poll attribute), 211  is_automatic_forward
InputVenueMessageContent (class in aiogram.types.input_venue_message_content), 276  INR (aiogram.enums.currency.Currency attribute), 476 intensity (aiogram.types.background_type_pattern.Background_t	attribute), 103  is_anonymous (aiogram.types.poll.Poll attribute), 211  is_automatic_forward
InputVenueMessageContent (class in aiogram.types.input_venue_message_content), 276  INR (aiogram.enums.currency.Currency attribute), 476 intensity (aiogram.types.background_type_pattern.Background_t	is_anonymous (aiogram.types.poll.Poll attribute), 211 is_automatic_forward
InputVenueMessageContent (class in aiogram.types.input_venue_message_content), 276  INR (aiogram.enums.currency.Currency attribute), 476 intensity (aiogram.types.background_type_pattern.Background_t	is_anonymous (aiogram.types.poll.Poll attribute), 211 is_automatic_forward
InputVenueMessageContent (class in aiogram.types.input_venue_message_content), 276  INR (aiogram.enums.currency.Currency attribute), 476 intensity (aiogram.types.background_type_pattern.Background_t	is_anonymous (aiogram.types.poll.Poll attribute), 211 is_automatic_forward
InputVenueMessageContent (class in aiogram.types.input_venue_message_content), 276  INR (aiogram.enums.currency.Currency attribute), 476 intensity (aiogram.types.background_type_pattern.Backgattribute), 24  INTERNAL_PASSPORT (aiogram.enums.encrypted_passport attribute), 478 invite_link (aiogram.methods.edit_chat_invite_link.Edit attribute), 341 invite_link (aiogram.methods.revoke_chat_invite_link.Edit attribute), 374 invite_link (aiogram.types.chat.Chat attribute), 37 invite_link (aiogram.types.chat_full_info.ChatFullInfo attribute), 59 invite_link (aiogram.types.chat_invite_link.ChatInviteL	is_anonymous (aiogram.types.poll.Poll attribute), 211 is_automatic_forward
InputVenueMessageContent (class in aiogram.types.input_venue_message_content), 276  INR (aiogram.enums.currency.Currency attribute), 476 intensity (aiogram.types.background_type_pattern.Backgattribute), 24  INTERNAL_PASSPORT (aiogram.enums.encrypted_passport attribute), 478 invite_link (aiogram.methods.edit_chat_invite_link.Editattribute), 341 invite_link (aiogram.methods.revoke_chat_invite_link.Editattribute), 374 invite_link (aiogram.types.chat.Chat attribute), 37 invite_link (aiogram.types.chat_full_info.ChatFullInfoattribute), 59 invite_link (aiogram.types.chat_invite_link.ChatInviteLattribute), 60	is_anonymous (aiogram.types.poll.Poll attribute), 211 is_automatic_forward
InputVenueMessageContent (class in aiogram.types.input_venue_message_content), 276  INR (aiogram.enums.currency.Currency attribute), 476 intensity (aiogram.types.background_type_pattern.Backgroute), 24  INTERNAL_PASSPORT (aiogram.enums.encrypted_passport attribute), 478 invite_link (aiogram.methods.edit_chat_invite_link.Edit attribute), 341 invite_link (aiogram.methods.revoke_chat_invite_link.Edit attribute), 374 invite_link (aiogram.types.chat_Chat attribute), 37 invite_link (aiogram.types.chat_full_info.ChatFullInfo attribute), 59 invite_link (aiogram.types.chat_invite_link.ChatInviteLattribute), 60 invite_link (aiogram.types.chat_join_request.ChatJoinEnter)	is_anonymous (aiogram.types.poll.Poll attribute), 211 is_automatic_forward
InputVenueMessageContent (class in aiogram.types.input_venue_message_content), 276  INR (aiogram.enums.currency.Currency attribute), 476 intensity (aiogram.types.background_type_pattern.Background_t	is_anonymous (aiogram.types.poll.Poll attribute), 211 is_automatic_forward
InputVenueMessageContent (class in aiogram.types.input_venue_message_content), 276  INR (aiogram.enums.currency.Currency attribute), 476 intensity (aiogram.types.background_type_pattern.Background_t	is_anonymous (aiogram.types.poll.Poll attribute), 211 is_automatic_forward
InputVenueMessageContent (class in aiogram.types.input_venue_message_content), 276  INR (aiogram.enums.currency.Currency attribute), 476 intensity (aiogram.types.background_type_pattern.Background_t	is_anonymous (aiogram.types.poll.Poll attribute), 211 is_automatic_forward
InputVenueMessageContent (class in aiogram.types.input_venue_message_content), 276  INR (aiogram.enums.currency.Currency attribute), 476 intensity (aiogram.types.background_type_pattern.Background_t	is_anonymous (aiogram.types.poll.Poll attribute), 211 is_automatic_forward
InputVenueMessageContent (class in aiogram.types.input_venue_message_content), 276  INR (aiogram.enums.currency.Currency attribute), 476 intensity (aiogram.types.background_type_pattern.Backgattribute), 24  INTERNAL_PASSPORT (aiogram.enums.encrypted_passport attribute), 478 invite_link (aiogram.methods.edit_chat_invite_link.Editattribute), 341 invite_link (aiogram.methods.revoke_chat_invite_link.Editattribute), 374 invite_link (aiogram.types.chat.Chat attribute), 37 invite_link (aiogram.types.chat_full_info.ChatFullInfoattribute), 59 invite_link (aiogram.types.chat_invite_link.ChatInviteLattribute), 60 invite_link (aiogram.types.chat_join_request.ChatJoinEattribute), 61 invite_link (aiogram.types.chat_member_updated.Chataattribute), 106 INVOICE (aiogram.enums.content_type.ContentType attribute), 474	is_anonymous (aiogram.types.poll.Poll attribute), 211 is_automatic_forward
InputVenueMessageContent (class in aiogram.types.input_venue_message_content), 276  INR (aiogram.enums.currency.Currency attribute), 476 intensity (aiogram.types.background_type_pattern.Background_t	is_anonymous (aiogram.types.poll.Poll attribute), 211 is_automatic_forward

is_forum (aiogram.types.chat_full_info.ChatFullInfo attribute), 58	keyboard (aiogram.types.reply_keyboard_markup.ReplyKeyboardMarkupattribute), 215
is_from_offline (aiogram.types.message.Message at-	
tribute), 159	aiogram.types.keyboard_button), 146
is_inverted(aiogram.types.background_type_pattern.Ba	
attribute), 24	aiogram.types.keyboard_button_poll_type),
is_local (aiogram.client.telegram.TelegramAPIServer	147
attribute), 14	KeyboardButtonPollTypeType (class in
is_manual (aiogram.types.text_quote.TextQuote at- tribute), 220	aiogram.enums.keyboard_button_poll_type_type), 480
is_member(aiogram.types.chat_member_restricted.ChatM.	A Knely by Character I was a line of the control of
attribute), 104	aiogram.types.keyboard_button_request_chat),
<pre>is_moving(aiogram.types.background_type_pattern.B</pre>	groundTypePeattern
attribute), 24	KeyboardButtonRequestUser (class in
is_moving (aiogram.types.background_type_wallpaper.Baattribute), 24	ackgroundT <b>ype§Yallptypers</b> .keyboard_button_request_user), 149
<pre>is_persistent(aiogram.types.reply_keyboard_markup.F</pre>	R <b>KelyKoalndBultLorRe</b> questUsers (class in
attribute), 215	aiogram.types.keyboard_button_request_users),
<pre>is_personal(aiogram.methods.answer_inline_query.Ans</pre>	werInlineQt@by
attribute), 449	KeyBuilder (class in aiogram.fsm.storage.base), 530
is_premium (aiogram.types.user.User attribute), 221	${\tt keywords}. \textit{SetStickerKeywords}. \textit{SetStickerKeywords}$
is_premium (aiogram.utils.web_app.WebAppUser	attribute), 314
attribute), 575	keywords (aiogram.types.input_sticker.InputSticker at-
$\verb"is_primary" (a iogram. types. chat_invite\_link. ChatInviteLink) is \verb"invite_link". ChatInviteLink is \verb"i$	nk tribute), 278
attribute), 60	KGS (aiogram.enums.currency.Currency attribute), 476
is_revoked(aiogram.types.chat_invite_link.ChatInviteLinattribute), 60	mKICKED (aiogram.enums.chat_member_status.ChatMemberStatus attribute), 472
<pre>is_topic_message (aiogram.types.message.Message</pre>	KRW (aiogram.enums.currency.Currency attribute), 476
attribute), 158	KZT (aiogram.enums.currency.Currency attribute), 476
is_unclaimed(aiogram.types.chat_boost_source_giveawattribute), 55	ay.ChatBoostSourceGiveaway L
is_video (aiogram.types.sticker.Sticker attribute), 279	label (aiogram.types.labeled_price.LabeledPrice
is_video (aiogram.types.sticker_set.StickerSet at-	attribute), 293
tribute), 281	LabeledPrice (class in aiogram.types.labeled_price),
ISK (aiogram.enums.currency.Currency attribute), 476	293
ITALIC (aiogram.enums.message_entity_type.MessageEn	it X Tanguage (aiogram.types.message_entity.MessageEntity
attribute), 481	attribute), 206
Italic (class in aiogram.utils.formatting), 585	language_code (aiogram.methods.delete_my_commands.DeleteMyCommattribute), 340
J	language_code (aiogram.methods.get_my_commands.GetMyCommands
JMD (aiogram.enums.currency.Currency attribute), 476	attribute), 357
<pre>join_by_request (aiogram.types.chat.Chat attribute), 38</pre>	language_code (aiogram.methods.get_my_description.GetMyDescription attribute), 359
<pre>join_by_request (aiogram.types.chat_full_info.ChatFull attribute), 59</pre>	Unfanguage_code (aiogram.methods.get_my_name.GetMyName attribute), 360
join_to_send_messages (aiogram.types.chat.Chat at- tribute), 38	language_code (aiogram.methods.get_my_short_description.GetMyShort attribute), 361
join_to_send_messages	language_code (aiogram.methods.set_my_commands.SetMyCommands
(aiogram.types.chat_full_info.ChatFullInfo	attribute), 421
attribute), 59	language_code (aiogram.methods.set_my_description.SetMyDescription
JPY (aiogram.enums.currency.Currency attribute), 476	attribute), 424
K	language_code (aiogram.methods.set_my_name.SetMyName attribute), 425
KES (aiogram.enums.currency.Currency attribute), 476	

language_code (aiogram.methods.set_my_short_descript attribute), 426	id <b>re£legtMh)(SiliogtDesmittinds</b> .send_video_note.SendVideoNote attribute), 407
language_code (aiogram.types.user.User attribute), 221 language_code (aiogram.utils.web_app.WebAppUser	length (aiogram.types.message_entity.MessageEntity attribute), 206
attribute), 575	length (aiogram.types.video_note.VideoNote attribute),
<pre>last_error_date(aiogram.types.webhook_info.Webhook</pre>	AInfo 227
attribute), 300	limit (aiogram.methods.get_updates.GetUpdates at-
<pre>last_error_message(aiogram.types.webhook_info.Webh</pre>	
attribute), 300	limit (aiogram.methods.get_user_profile_photos.GetUserProfilePhotos
<pre>last_name (aiogram.methods.send_contact.SendContact</pre>	attribute), 363
attribute), 382	link_preview_options
last_name (aiogram.types.chat.Chat attribute), 35	(aiogram.methods.edit_message_text.EditMessageText
last_name (aiogram.types.chat_full_info.ChatFullInfo	attribute), 444
attribute), 58	link_preview_options
<pre>last_name (aiogram.types.contact.Contact attribute),</pre>	(aiogram.methods.send_message.SendMessage
127	attribute), 394
<pre>last_name (aiogram.types.inline_query_result_contact.In.</pre>	
attribute), 252	(aiogram.types.external_reply_info.ExternalReplyInfo
<pre>last_name (aiogram.types.input_contact_message_conten</pre>	
attribute), 271	link_preview_options
<pre>last_name (aiogram.types.shared_user.SharedUser at-</pre>	(aiogram.types.input_text_message_content.InputTextMessageCo
tribute), 218	attribute), 276
last_name (aiogram.types.user.User attribute), 221	link_preview_options
last_name (aiogram.utils.web_app.WebAppUser at-	(aiogram.types.message.Message attribute),
tribute), 575	159
last_synchronization_error_date	linked_chat_id (aiogram.types.chat.Chat attribute),
(aiogram.types.webhook_info.WebhookInfo	38
attribute), 300	linked_chat_id(aiogram.types.chat_full_info.ChatFullInfo
latitude (aiogram.methods.edit_message_live_location.E	
attribute), 439	LinkPreviewOptions (class in
latitude (aiogram.methods.send_location.SendLocation	aiogram.types.link_preview_options), 151
attribute), 389	live_period(aiogram.methods.edit_message_live_location.EditMessage1
latitude (aiogram.methods.send_venue.SendVenue at-	attribute), 439
tribute), 401	live_period(aiogram.methods.send_location.SendLocation
latitude (aiogram.types.inline_query_result_location.Inl	
attribute), 259	live_period(aiogram.types.inline_query_result_location.InlineQueryRes
latitude (aiogram.types.inline_query_result_venue.Inline	
	live_period(aiogram.types.input_location_message_content.InputLocati
latitude (aiogram.types.input_location_message_content	
attribute), 274	live_period (aiogram.types.location.Location at-
latitude (aiogram.types.input_venue_message_content.Ir	- · · · · · · · · · · · · · · · · · · ·
attribute), 276	LKR (aiogram.enums.currency.Currency attribute), 476
latitude (aiogram.types.location.Location attribute),	LOCATION (aiogram.enums.content_type.ContentType at-
152	tribute), 473
LBP (aiogram.enums.currency.Currency attribute), 476	LOCATION (aiogram.enums.inline_query_result_type.InlineQueryResultTyp
leave() (aiogram.fsm.scene.SceneWizard method), 548	attribute), 479
<pre>leave() (aiogram.types.chat.Chat method), 44</pre>	location(aiogram.types.business_location.BusinessLocation
LeaveChat (class in aiogram.methods.leave_chat), 364	attribute), 31
LEFT (aiogram.enums.chat_member_status.ChatMemberSt	aluocation (aiogram.types.chat.Chat attribute), 38
attribute), 472	location (aiogram.types.chat_full_info.ChatFullInfo at-
LEFT_CHAT_MEMBER (aiogram.enums.content_type.Conten	
attribute), 474	location (aiogram.types.chat_location.ChatLocation
<pre>left_chat_member (aiogram.types.message.Message</pre>	attribute), 99
attribute), 160	location(aiogram.types.chosen inline result.ChosenInlineResult

attribute), 229	MaskPos	sitionPoint	(class	in
location (aiogram.types.external_reply_info.ExternalRep attribute), 130		aiogram.enums.mask_p nections(aiogram.me		
location (aiogram.types.inline_query.InlineQuery at- tribute), 230	max_con	attribute), 468 nections (aiogram.typ	es.webhook_info.V	VebhookInfo
location (aiogram.types.message.Message attribute), 160	max_qua	attribute), 301 antity (aiogram.types.k	eyboard_button_r	equest_users.Keyboard
location (aiogram.types.venue.Venue attribute), 224		attribute), 150		
Location (class in aiogram.types.location), 152		ction_count(aiogram	ı.types.chat_full_ir	nfo.ChatFullInfo
login_url (aiogram.types.inline_keyboard_button.Inlinekattribute), 138		_amount (aiogram.met)	hods.create_invoic	e_link.CreateInvoiceLi
LoginUrl (class in aiogram.types.login_url), 153		attribute), 459		
LogOut (class in aiogram.methods.log_out), 365		_amount (aiogram.meth	hods.send_invoice.	SendInvoice
longitude (aiogram.methods.edit_message_live_location.		_		
attribute), 439		_amount (aiogram.type	s.input_invoice_m	essage_content.InputIn
longitude (aiogram.methods.send_location.SendLocation		attribute), 273		
attribute), 389	MaybeIn	accessibleMessage	(class	in
longitude (aiogram.methods.send_venue.SendVenue attribute), 402		aiogram.types.maybe_i 153		
longitude (aiogram.types.inline_query_result_location.In attribute), 259	a <b>hinde Quees</b> rț	y <b>Réxilt&amp;xvattyp</b> es.messo 163	age.Message pro	perty),
$longitude ({\it aiogram.types.inline\_query\_result\_venue.Inline\_query\_r$	ı <b>MQ</b> Lıe(reyiR	<b>gsahtVemwe</b> rs.currency.Ci	urrency attribute),	476
attribute), 265	media(a	iogram.methods.edit_m	essage_media.Edii	MessageMedia
longitude (aiogram.types.input_location_message_contentatribute), 274	_	c <b>atticihMe}</b> s&&Content iogram.methods.send_n	nedia_group.SendI	MediaGroup
longitude (aiogram.types.input_venue_message_content.n	InputVenu	e <b>Miribauge</b> )ÇƏhitlent		
attribute), 276	media(a	iogram.types.input_mea	lia_animation.Inpi	ıtMediaAnimation
longitude (aiogram.types.location.Location attribute),		attribute), 140		
152	media(a	iogram.types.input_mea attribute), 142	lia_audio.InputMe	diaAudio
M	media(a	iogram.types.input_mea	lia_document.Inpu	tMediaDocument
MAD (aiogram.enums.currency.Currency attribute), 476		attribute), 143		
<pre>magic_data (aiogram.filters.magic_data.MagicData at- tribute), 506</pre>	media(a	iogram.types.input_mea attribute), 144	lia_photo.InputMe	diaPhoto
magic_result(aiogram.filters.command.CommandObjectattribute), 500	tmedia(a	iogram.types.input_mea attribute), 145	lia_video.InputMe	diaVideo
MagicData (class in aiogram.filters.magic_data), 506 make_request() (aiogram.client.session.base.BaseSessio		roup_id (aiogram.typ tribute), 159	es.message.Messa	ge at-
method), 14		oupBuilder	(class	in
MARKDOWN (aiogram.enums.parse_mode.ParseMode at-		aiogram.utils.media_g	roup), 587	
tribute), 482	MEMBER (	aiogram.enums.chat_m	ember_status.Chat	MemberStatus
MARKDOWN_V2 (aiogram.enums.parse_mode.ParseMode		attribute), 472		
attribute), 482	member_	limit(aiogram.method	ls.create_chat_inv	ite_link.CreateChatInvi
MASK (aiogram.enums.sticker_type.StickerType attribute),		attribute), 333		
483	member_	limit (aiogram.method	ls.edit chat invite	link.EditChatInviteLir
mask_position(aiogram.methods.set_sticker_mask_posit				
attribute), 315		limit(aiogram.types.c	hat_invite_link.Ch	atInviteLink
mask_position(aiogram.types.input_sticker.InputSticker		attribute), 61		
attribute), 278	member_	status_changed		
mask_position (aiogram.types.sticker.Sticker attribute),	_	(aiogram.filters.chat_n	nember_updated.C	hatMemberUpdatedFil
279		attribute), 501	— <b>1</b>	
MaskPosition (class in aiogram.types.mask_position),	MemoryS	storage (class in aiogra	am.fsm.storage.me	mory),
278	•	529	. 0	• •
<del>-</del> · •	MENTION	l (aiogram enums messa	oe entity tyne Me	ssageEntityTyne

 ${\tt mention}\ (a iogram. filters. command. Command Object\ at-$ 

mention\_html() (aiogram.types.user.User method),

(aiogram.types.user.User

attribute), 481

tribute), 500

*method*), 221

221

mention\_markdown()

${\tt mentioned}  (a iogram. filters. command. Command Object$	(aiogram.types.message.Message attribute),
property), 500	161
	MbssMgpuBdt(aiogram.methods.copy_message.CopyMessage
attribute), 414	attribute), 329
	message_id(aiogram.methods.delete_message.DeleteMessage
MenuButtonCommands (class in	attribute), 435
	${\tt message\_id} (\it aiogram.methods.edit\_message\_caption.EditMessageCaption.editMessageCa$
154	attribute), 437
	${\tt message\_id} (\it aiogram.methods.edit\_message\_live\_location. EditMessageLi$
aiogram.types.menu_button_default), 155	attribute), 439
	message_id(aiogram.methods.edit_message_media.EditMessageMedia
aiogram.enums.menu_button_type), 481	attribute), 441
	message_id(aiogram.methods.edit_message_reply_markup.EditMessageR
aiogram.types.menu_button_web_app), 155	attribute), 442
	message_id(aiogram.methods.edit_message_text.EditMessageText
tribute), 484	attribute), 444
	https://www.methods.forward_message.ForwardMessage
property), 558	attribute), 345
message (alogram.types.business_intro.BusinessIntro at- tribute), 31	<pre>message_id(aiogram.methods.get_game_high_scores.GetGameHighScore</pre>
	message_id(aiogram.methods.pin_chat_message.PinChatMessage
attribute), 33	attribute), 366
${\tt message} \ (aiogram.types.passport\_element\_error\_data\_field)$	th <b>Esspor</b> t <b>ElectuingFarmnDettuldicke</b> t_game_score.SetGameScore
attribute), 285	attribute), 455
	message_reaction.SetMessageReaction
attribute), 285	attribute), 419
message (aiogram.types.passport_element_error_files.Pass) attribute), 286	messEbgneritE(wirrFiders.methods.stop_message_live_location.StopMessageLi attribute), 445
message (aiogram.types.passport_element_error_front_side	
attribute), 287	attribute), 447
	ridesBaspoitElaingniFirmeRodersaSpile_chat_message.UnpinChatMessage
attribute), 288	attribute), 433
	Mass Etgeneix Earwa Evelfitypes.external_reply_info.ExternalReplyInfo
attribute), 289	attribute), 129
	meskaBesspo (uliberrant Spess Translatiibheilaessage.Inaccessible Message
attribute), 290	attribute), 137
message (aiogram.types.passport_element_error_translatio attribute), 291	mesikasgleasispooniblemmentEpresorileasuskatiMnesiluse attribute),  158
message (aiogram.types.passport_element_error_unspecific	
attribute), 291	tribute), 206
message (aiogram.types.update.Update attribute), 298 Message (class in aiogram.types.message), 156	<pre>message_id(aiogram.types.message_origin_channel.MessageOriginChannel.tribute), 207</pre>
	message_id(aiogram.types.message_reaction_count_updated.MessageReactitibute), 209
	message_id(aiogram.types.message_reaction_updated.MessageReactionU
(aiogram.types.chat_full_info.ChatFullInfo	attribute), 210
	message_id(aiogram.types.reply_parameters.ReplyParameters

message\_auto\_delete\_time

attribute), 205

attribute), 474

 ${\tt MESSAGE\_AUTO\_DELETE\_TIMER\_CHANGED}$ 

message\_auto\_delete\_timer\_changed

(aiogram.enums.content\_type.ContentType

(aiogram.types.message\_auto\_delete\_timer\_changed.MessageAut

attribute), 217	attribute), 389		
message_ids(aiogram.methods.copy_messages.CopyMessmgssage		nethods.send_media_	group.SendMediaGr
attribute), 331	attribute), 391		- C IM
message_ids(aiogram.methods.delete_messages.DeleteMenassage		netnoas.sena_messag	e.Senamessage
attribute), 436	attribute), 394	wathada aand mhata C	and Dhata
message_ids(aiogram.methods.forward_messages.Forwardlessage	geschreau_1u (aiogram.r attribute), 396	neinoas.sena_pnoio. <b>s</b>	enarnoio
attribute), 347 message_ids (aiogram.types.business_messages_deleted.Business_deleted.Busin		mathada sand nall Car	ad <b>D</b> all
attribute), 32	attribute), 399	neinoas.sena_poii.sei	шғош
MESSAGE_REACTION (aiogram.enums.update_type.UpdateTypessage	**	nothods sond sticker	SondStickor
attribute), 484	attribute), 311	петоиз.зени_зискет.	Jenusiicker
message_reaction (aiogram.types.update.Update at- message	, .	nethods.send_venue.S	SendVenue
tribute), 299	attribute), 402		
	e_thread_id( <i>aiogram.r</i>	nethods.send video.S	endVideo
(aiogram.enums.update_type.UpdateType	attribute), 404		
	e_thread_id( <i>aiogram.r</i>	nethods.send video r	ote.SendVideoNote
message_reaction_count	attribute), 407		
(aiogram.types.update.Update attribute), message	e_thread_id( <i>aiogram.r</i>	nethods.send_voice.Se	endVoice
299	attribute), 410		
message_text(aiogram.types.input_text_message_contentrilepsidign	<u>xtM<b>hr</b>æ</u> gd <u>C</u> iodi(eniogram.1	nethods.unpin_all_for	rum_topic_messages.
attribute), 276	attribute), 431		
${\tt message\_thread\_id} \ (aiogram.methods.close\_forum\_topicn {\tt Colored}) \\ {\tt message\_thread\_i$	<b>e<u>ru</u>tchFæpæid_</b> id(aiogram.t	ypes.forum_topic.For	umTopic
attribute), 327	attribute), 132		
$\verb message_thread_id   (a iogram.methods.copy\_message.Copy   \verb dsage   desage   des$	<u>ge</u> thread_id ( <i>aiogram</i>	types.message.Messa	ige
attribute), 329	attribute), 158		
$\verb message_thread_id   (a iogram.methods.copy\_messages.Copp \verb messages   A copp \verb messages   (a iogram.methods.copy\_messages.Copp \verb messages   (a iogram.methods.copy\_messages.copy\_messages.Copp \verb messages   (a iogram.methods.copy\_messages.copy\_messages.copy\_messages.copy\_messages.copy\_messages.copy\_messages.copy\_messages.copy\_messages.copy\_messages.copy\_messages.copy\_messages.copy\_messages.copy\_messages.copy\_messages.copy\_messages.copy\_messa$			in
attribute), 331	aiogram.types.message	_auto_delete_timer_c	hanged),
${\tt message\_thread\_id} (\it aiogram.methods.delete\_forum\_topic.Delete\_f$	_		
	•	(class	in
$\verb message_thread_id   (a iogram.methods.edit\_forum\_topic.EditForum\_topic.Edi$			
	eEntityType	(class	in
message_thread_id(aiogram.methods.forward_message.Forward			
	eId (class in aiogram.typ	-	
message_thread_id(aiogram.methods.forward_messagesMessage	- 0	(class	in
attribute), 347	aiogram.types.message		·
message_thread_id(aiogram.methods.reopen_forum_topNessage	- *	(class	in
attribute), 370	aiogram.types.message	_origin_cnannei),	
message_thread_id(aiogram.methods.send_animation.SendAnim		(alaaa	·
	eOriginChat	(class	in
<pre>message_thread_id(aiogram.methods.send_audio.SendAudio</pre>	<i>aiogram.types.message</i> eOriginHiddenUser		in
message_thread_id(aiogram.methods.send_chat_action.SendCha	3	(class	in
attribute), 381	1208	_origin_niaden_user)	),
message_thread_id(aiogram.methods.send_contact.SendMessage		(class	in
attribute), 382	aiogram.enums.messag		iii
message_thread_id(aiogram.methods.send_dice.SendDidlessage		(class	in
attribute), 384	aiogram.types.message		111
message_thread_id(aiogram.methods.send_document.SeMdDsagn			in
attribute), 387	aiogram.types.message		
message_thread_id(aiogram.methods.send_game.SendGame	209		····· · · · · · · · · · · · · · · · ·
	eReactionUpdated	(class	in
message_thread_id(aiogram.methods.send_invoice.SendInvoice	aiogram.types.message	*	
attribute), 462	210	= <b>1</b> //	
message_thread_id(aiogram.methods.send_location.SendMLGRATA	FROM_CHAT_ID		

```
(aiogram.enums.content_type.ContentType
                                                               (aiogram.methods.close.Close
                                                                                                 attribute),
        attribute), 474
                                                               326
migrate_from_chat_id
                                                      model_computed_fields
                                                               (aiogram.methods.close_forum_topic.CloseForumTopic
         (aiogram.types.message.Message
                                          attribute),
                                                               attribute), 327
MIGRATE_TO_CHAT_ID (aiogram.enums.content type.Contemofleek_computed_fields
         attribute), 474
                                                               (aiogram.methods.close general forum topic.CloseGeneralForu
migrate_to_chat_id (aiogram.types.message.Message
                                                               attribute), 328
         attribute), 161
                                                      model_computed_fields
migrate_to_chat_id(aiogram.types.response_parameters.ResponseMagmatamethods.copy_message.CopyMessage
         attribute), 218
                                                               attribute), 330
                                                     model_computed_fields
mime_type (aiogram.types.animation.Animation
                                                               (aiogram.methods.copy_messages.CopyMessages
        tribute), 19
mime_type (aiogram.types.audio.Audio attribute), 20
                                                               attribute), 332
             (aiogram.types.document.Document
                                                 at- model_computed_fields
mime_type
         tribute), 128
                                                               (aiogram.methods.create_chat_invite_link.CreateChatInviteLink
mime_type (aiogram.types.inline_query_result_document.InlineQuery_ReibultDocument
                                                      model_computed_fields
         attribute), 255
mime_type (aiogram.types.inline_query_result_video.InlineQueryResultivigidam.methods.create_forum_topic.CreateForumTopic
                                                               attribute), 334
         attribute), 267
mime_type (aiogram.types.video.Video attribute), 225
                                                      model_computed_fields
mime_type (aiogram.types.voice.Voice attribute), 228
                                                               (aiogram.methods.create_invoice_link.CreateInvoiceLink
MNT (aiogram.enums.currency.Currency attribute), 476
                                                               attribute), 460
model_computed_fields
                                                      model_computed_fields
         (aiogram.filters.callback_data.CallbackData
                                                               (aiogram.methods.create_new_sticker_set.CreateNewStickerSet
        attribute), 507
                                                               attribute), 304
model_computed_fields
                                                      model_computed_fields
         (aiogram.methods.add_sticker_to_set.AddStickerToSet
                                                               (aiogram.methods.decline_chat_join_request.DeclineChatJoinRe
        attribute), 303
                                                               attribute), 335
model_computed_fields
                                                      model_computed_fields
         (aiogram.methods.answer_callback_query.AnswerCallbackQuiogram.methods.delete_chat_photo.DeleteChatPhoto
        attribute), 321
                                                               attribute), 336
model_computed_fields
                                                      model_computed_fields
         (aiogram.methods.answer_inline_query.AnswerInlineQuery(aiogram.methods.delete_chat_sticker_set.DeleteChatStickerSet
         attribute), 449
                                                               attribute), 337
model_computed_fields
                                                      model_computed_fields
         (aiogram.methods.answer pre checkout query.AnswerPreClaiokoantOmmethods.delete forum topic.DeleteForumTopic
         attribute), 456
                                                               attribute), 338
model_computed_fields
                                                      model_computed_fields
         (aiogram.methods.answer_shipping_query.AnswerShipping_Quegram.methods.delete_message.DeleteMessage
         attribute), 457
                                                               attribute), 435
model_computed_fields
                                                      model_computed_fields
         (aiogram.methods.answer_web_app_query.AnswerWebAppQuuegram.methods.delete_messages.DeleteMessages
         attribute), 450
                                                               attribute), 436
model_computed_fields
                                                      model_computed_fields
         (aiogram.methods.approve_chat_join_request.ApproveChat.lainReamestethods.delete_my_commands.DeleteMyCommands
        attribute), 322
                                                               attribute), 340
model_computed_fields
                                                      model_computed_fields
         (aiogram.methods.ban_chat_member.BanChatMember
                                                               (aiogram.methods.delete\_sticker\_from\_set.DeleteStickerFromSet
        attribute), 323
                                                               attribute), 305
model_computed_fields
                                                      model_computed_fields
        (aiogram.methods.ban_chat_sender_chat.BanChatSenderChatogram.methods.delete_sticker_set.DeleteStickerSet
        attribute), 325
                                                               attribute), 306
model_computed_fields
                                                      model_computed_fields
```

```
(aiogram.methods.delete webhook.DeleteWebhook
                                                                                                     (aiogram.methods.get_custom_emoji_stickers.GetCustomEmojiSt
              attribute), 464
                                                                                                     attribute), 307
model_computed_fields
                                                                                      model_computed_fields
              (aiogram.methods.edit_chat_invite_link.EditChatInviteLink (aiogram.methods.get_file.GetFile
                                                                                                                                                          attribute),
              attribute), 341
                                                                                                     354
model_computed_fields
                                                                                      model_computed_fields
              (aiogram.methods.edit forum topic.EditForumTopic
                                                                                                    (aiogram.methods.get forum topic icon stickers.GetForumTopic
              attribute), 342
                                                                                                     attribute), 355
model_computed_fields
                                                                                      model_computed_fields
              (aiogram.methods.edit\_general\_forum\_topic.EditGeneralFo \cite{Content} and \cite{Conten
              attribute), 343
                                                                                                    attribute), 452
model_computed_fields
                                                                                      model_computed_fields
              (aiogram.methods.edit_message_caption.EditMessageCaptioniogram.methods.get_me.GetMe
                                                                                                                                                          attribute),
              attribute), 437
                                                                                                     356
model_computed_fields
                                                                                      model_computed_fields
              (aiogram.methods.edit_message_live_location.EditMessageLive_twotinethods.get_my_commands.GetMyCommands
              attribute), 439
                                                                                                    attribute), 357
                                                                                      model_computed_fields
model_computed_fields
              (aiogram.methods.edit_message_media.EditMessageMedia (aiogram.methods.get_my_default_administrator_rights.GetMyDo
                                                                                                    attribute), 358
              attribute), 441
model_computed_fields
                                                                                      model_computed_fields
              (aiogram.methods.edit_message_reply_markup.EditMessageRuptyMurkethods.get_my_description.GetMyDescription
              attribute), 442
                                                                                                     attribute), 359
model_computed_fields
                                                                                      model_computed_fields
              (aiogram.methods.edit_message_text.EditMessageText
                                                                                                    (aiogram.methods.get_my_name.GetMyName
              attribute), 444
                                                                                                    attribute), 360
model_computed_fields
                                                                                      model_computed_fields
              (aiogram.methods.export_chat_invite_link.ExportChatInvite/laiiolgram.methods.get_my_short_description.GetMyShortDescrip
              attribute), 344
                                                                                                    attribute), 361
model_computed_fields
                                                                                      model_computed_fields
              (aiogram.methods.forward_message.ForwardMessage
                                                                                                    (aiogram.methods.get_sticker_set.GetStickerSet
              attribute), 346
                                                                                                     attribute), 308
model_computed_fields
                                                                                      model_computed_fields
              (aiogram.methods.forward_messages.ForwardMessages
                                                                                                    (aiogram.methods.get_updates.GetUpdates
              attribute), 347
                                                                                                    attribute), 466
model_computed_fields
                                                                                      model_computed_fields
              (aiogram.methods.get_business_connection.GetBusinessConnection.methods.get_user_chat_boosts.GetUserChatBoosts
              attribute), 348
                                                                                                    attribute), 362
model_computed_fields
                                                                                      model_computed_fields
              (aiogram.methods.get_chat.GetChat attribute),
                                                                                                    (aiogram.methods.get_user_profile_photos.GetUserProfilePhotos
                                                                                                    attribute), 362
model_computed_fields
                                                                                      model_computed_fields
              (aiogram.methods.get chat administrators.GetChatAdministratiovs.methods.get webhook info.GetWebhookInfo
              attribute), 350
                                                                                                    attribute), 467
model_computed_fields
                                                                                      model_computed_fields
              (aiogram.methods.get_chat_member.GetChatMember
                                                                                                    (aiogram.methods.hide_general_forum_topic.HideGeneralForum
              attribute), 351
                                                                                                    attribute), 363
model_computed_fields
                                                                                      model_computed_fields
              (aiogram.methods.get_chat_member_count.GetChatMembet@ingrtim.methods.leave_chat.LeaveChat
              attribute), 352
                                                                                                    attribute), 364
model_computed_fields
                                                                                      model_computed_fields
              (aiogram.methods.get chat menu button.GetChatMenuBut(aniogram.methods.log out.LogOut attribute),
              attribute), 353
model_computed_fields
                                                                                      model_computed_fields
```

	(aiogram.methods.pin_chat_message.PinChatMe attribute), 366	ssage	(aiogram.methods.send_photo.SendPhoto attribute), 396	
model_	computed_fields	model_c	computed_fields	
	(aiogram.methods.promote_chat_member.Promotattribute), 368			
model	_computed_fields	model d	computed_fields	
	(aiogram.methods.reopen_forum_topic.ReopenFo		=	
	attribute), 370		attribute), 311	
model	_computed_fields	model d	computed_fields	
	(aiogram.methods.reopen_general_forum_topic.I		•	
	attribute), 371	1	attribute), 402	
model_	_computed_fields	model_c	computed_fields	
	(aiogram.methods.replace_sticker_in_set.Replace	eStickerIn	Seaiogram.methods.send_video.SendVideo	
	attribute), 309		attribute), 405	
model_	_computed_fields	model_c	computed_fields	
	(aiogram.methods.restrict_chat_member.Restrict	ChatMem	beniogram.methods.send_video_note.SendVideoNote	
	attribute), 372		attribute), 407	
model_	computed_fields	model_c	computed_fields	
	(aiogram.methods.revoke_chat_invite_link.Revok	eChatInvi	it <b>&amp;Lin</b> &ram.methods.send_voice.SendVoice	
	attribute), 374		attribute), 410	
model_	_computed_fields	model_c	computed_fields	
	(aiogram.methods.send_animation.SendAnimatio	n	(aiogram.methods.set_chat_administrator_custom_title.S	SetChatA
	attribute), 376		attribute), 412	
model_	_computed_fields	model_c	computed_fields	
	$(aiogram.methods.send\_audio.SendAudio$		$(aiogram.methods.set\_chat\_description.SetChatDescription) \\$	ion
	attribute), 379		attribute), 413	
model_	computed_fields	model_c	computed_fields	
	(aiogram.methods.send_chat_action.SendChatAcattribute), 381	tion	(aiogram.methods.set_chat_menu_button.SetChatMenuBattribute), 414	lutton
model_	_computed_fields	model_c	computed_fields	
	(aiogram.methods.send_contact.SendContact		(aiogram.methods.set_chat_permissions.SetChatPermissi	ions
	attribute), 382		attribute), 415	
model_	computed_fields	model_c	computed_fields	
	(aiogram.methods.send_dice.SendDice at-		(aiogram.methods.set_chat_photo.SetChatPhoto	
	tribute), 385		attribute), 416	
model_	.computed_fields	model_c	computed_fields	
	(aiogram.methods.send_document.SendDocumen	t	$(aiogram.methods.set\_chat\_sticker\_set.SetChatStickerSet$	t
	attribute), 387		attribute), 417	
$model_{-}$	_computed_fields	model_c	computed_fields	
	$(aiogram.methods.send\_game.SendGame$		(aiogram.methods.set_chat_title.SetChatTitle	
	attribute), 453		attribute), 418	
model_	computed_fields	model_c	computed_fields	
	$(aiogram.methods.send\_invoice.SendInvoice$		(aiogram.methods.set_custom_emoji_sticker_set_thumbn	ail.SetC
	attribute), 462		attribute), 312	
model_	_computed_fields	model_c	computed_fields	
	(aiogram.methods.send_location.SendLocation attribute), 389		(aiogram.methods.set_game_score.SetGameScore attribute), 455	
$model_{-}$	_computed_fields	model_c	computed_fields	
	(aiogram.methods.send_media_group.SendMedia	Group	$(a iogram. methods. set\_message\_reaction. Set Message Reaction and the set of the set $	ction
	attribute), 392		attribute), 419	
model_	_computed_fields	model_c	computed_fields	
	(aiogram.methods.send_message.SendMessage		(aiogram.methods.set_my_commands.SetMyCommands	
	attribute), 394		attribute), 421	
model	_computed_fields	model c	computed_fields	

```
(aiogram.methods.set_my_default_administrator_rights.SetMsiDefaultAdthirdstwapinRightsforum_topic_messages.UnpinAllFor
        attribute), 422
                                                               attribute), 431
model_computed_fields
                                                      model_computed_fields
         (aiogram.methods.set_my_description.SetMyDescription
                                                               (aiogram.methods.unpin_all_general_forum_topic_messages.Unp
         attribute), 423
                                                               attribute), 432
model_computed_fields
                                                      model_computed_fields
         (aiogram.methods.set_my_name.SetMyName
                                                               (aiogram.methods.unpin_chat_message.UnpinChatMessage
         attribute), 424
                                                               attribute), 433
model_computed_fields
                                                      model_computed_fields
         (aiogram.methods.set_my_short_description.SetMyShortDesaingtam.methods.upload_sticker_file.UploadStickerFile
         attribute), 425
                                                               attribute), 320
model_computed_fields
                                                      model_computed_fields
         (aiogram.methods.set_passport_data_errors.SetPassportDataErgoxm.types.animation.Animation attribute),
        attribute), 470
model_computed_fields
                                                      model_computed_fields (aiogram.types.audio.Audio
         (aiogram.methods.set_sticker_emoji_list.SetStickerEmojiListattribute), 20
        attribute), 313
                                                      model_computed_fields
model_computed_fields
                                                               (aiogram.types.background_fill.BackgroundFill
         (aiogram.methods.set_sticker_keywords.SetStickerKeywordsattribute), 20
                                                      model_computed_fields
         attribute), 314
model_computed_fields
                                                               (aiogram.types.background_fill_freeform_gradient.BackgroundFi
         (aiogram.methods.set_sticker_mask_position.SetStickerMaskaPoribinoen), 21
         attribute), 315
                                                      model_computed_fields
model_computed_fields
                                                               (aiogram.types.background fill gradient.BackgroundFillGradien
         (aiogram.methods.set_sticker_position_in_set.SetStickerPosititnibuse), 21
        attribute), 316
                                                      model_computed_fields
model_computed_fields
                                                               (aiogram.types.background_fill_solid.BackgroundFillSolid
         (aiogram.methods.set_sticker_set_thumbnail.SetStickerSetThutmlbnatel), 22
                                                      model_computed_fields
        attribute), 317
model_computed_fields
                                                               (aiogram.types.background_type.BackgroundType
         (aiogram.methods.set_sticker_set_title.SetStickerSetTitle
                                                               attribute), 22
        attribute), 319
                                                      model_computed_fields
model_computed_fields
                                                               (aiogram.types.background_type_chat_theme.BackgroundTypeCh
         (aiogram.methods.set_webhook.SetWebhook
                                                               attribute), 22
         attribute), 468
                                                      model_computed_fields
model_computed_fields
                                                               (aiogram.types.background_type_fill.BackgroundTypeFill
         (aiogram.methods.stop_message_live_location.StopMessageditivelbuteation)
         attribute), 446
                                                      model_computed_fields
model_computed_fields
                                                               (aiogram.types.background_type_pattern.BackgroundTypePattern
         (aiogram.methods.stop_poll.StopPoll attribute),
                                                               attribute), 23
                                                      model_computed_fields
model_computed_fields
                                                               (aiogram.types.background_type_wallpaper.BackgroundTypeWall
         (aiogram.methods.unban_chat_member.UnbanChatMemberattribute), 24
         attribute), 427
                                                      model_computed_fields
model_computed_fields
                                                               (aiogram.types.birthdate.Birthdate attribute),
         (aiogram.methods.unban_chat_sender_chat.UnbanChatSend&rChat
        attribute), 428
                                                      model_computed_fields
model_computed_fields
                                                               (aiogram.types.bot_command.BotCommand
         (aiogram.methods.unhide_general_forum_topic.UnhideGenentaliBorta)n; Zōpic
        attribute), 429
                                                      model_computed_fields
model_computed_fields
                                                               (aiogram.types.bot_command_scope.BotCommandScope
        (aiogram.methods.unpin all chat messages.UnpinAllChatMutssibges), 26
                                                      model_computed_fields
        attribute), 430
model_computed_fields
                                                               (aiogram.types.bot command scope all chat administrators.Bo
```

	attribute), 26	model_computed_fields
mode1_	computed_fields	(aiogram.types.chat_administrator_rights.ChatAdministratorRight
	(aiogram.types.bot_command_scope_all_group_	
model	<pre>attribute), 27 computed_fields</pre>	model_computed_fields
mode1_	_	(aiogram.types.chat_background.ChatBackground
	(aiogram.types.bot_command_scope_all_private_attribute), 27	_cnais.Boi@www.gsaapeAnFrivateChais model_computed_fields
modol	computed_fields	(aiogram.types.chat_boost.ChatBoost at-
mode1_	(aiogram.types.bot_command_scope_chat.BotCo	
	attribute), 27	model_computed_fields
model	computed_fields	(aiogram.types.chat_boost_added.ChatBoostAdded
moder_	(aiogram.types.bot_command_scope_chat_admir	
	attribute), 28	model_computed_fields
model	computed_fields	(aiogram.types.chat_boost_removed.ChatBoostRemoved
mouci_	(aiogram.types.bot_command_scope_chat_memb	
	attribute), 29	model_computed_fields
model	computed_fields	(aiogram.types.chat_boost_source.ChatBoostSource
mouci_	(aiogram.types.bot_command_scope_default.Bot)	
	attribute), 29	model_computed_fields
model	computed_fields	(aiogram.types.chat_boost_source_gift_code.ChatBoostSourceGi
mouci_	(aiogram.types.bot_description.BotDescription	attribute), 54
	attribute), 29	model_computed_fields
model	computed_fields	(aiogram.types.chat_boost_source_giveaway.ChatBoostSourceGi
mouci_	(aiogram.types.bot_name.BotName attribute),	attribute), 55
	30	model_computed_fields
model	computed_fields	(aiogram.types.chat_boost_source_premium.ChatBoostSourcePre
	(aiogram.types.bot_short_description.BotShortDe	
	attribute), 30	model_computed_fields
model_	computed_fields	(aiogram.types.chat_boost_updated.ChatBoostUpdated
	(aiogram.types.business_connection.BusinessCon	
	attribute), 30	model_computed_fields
model_	computed_fields	(aiogram.types.chat_full_info.ChatFullInfo
	(aiogram.types.business_intro.BusinessIntro	attribute), 58
	attribute), 31	model_computed_fields
model_	computed_fields	(aiogram.types.chat_invite_link.ChatInviteLink
	(aiogram.types.business_location.BusinessLocati	
	attribute), 31	model_computed_fields
model_	computed_fields	(aiogram.types.chat_join_request.ChatJoinRequest
	(aiogram.types.business_messages_deleted.Busin	nessMessag <b>asDidinte</b> ), 93
	attribute), 32	model_computed_fields
model_	computed_fields	(aiogram.types.chat_location.ChatLocation
	(aiogram.types.business_opening_hours.Business	sOpeningH <b>attri</b> bute), 99
	attribute), 32	model_computed_fields
model_	computed_fields	(aiogram.types.chat_member.ChatMember
	(aiogram.types.business_opening_hours_interval	l.Business <b>OpteriIngEl</b> Q@FsInterval
	attribute), 33	model_computed_fields
model_	computed_fields	(aiogram.types.chat_member_administrator.ChatMemberAdminis
	(aiogram.types.callback_game.CallbackGame	attribute), 101
	attribute), 301	model_computed_fields
model_	computed_fields	(aiogram.types.chat_member_banned.ChatMemberBanned
	(aiogram.types.callback_query.CallbackQuery	attribute), 102
	attribute), 34	model_computed_fields
model_	<pre>computed_fields (aiogram.types.chat.Chat at-</pre>	(aiogram.types.chat_member_left.ChatMemberLeft
	tribute), 38	attribute), 102

model_computed_fields	attribute), 132
(aiogram.types.chat_member_member.ChatMen	mb <b>moldel</b> nbeomputed_fields
attribute), 103	(aiogram.types.forum_topic_created.ForumTopicCreated
model_computed_fields	attribute), 133
(aiogram.types.chat_member_owner.ChatMemb	
attribute), 103	(aiogram.types.forum_topic_edited.ForumTopicEdited
model_computed_fields	attribute), 133
(aiogram.types.chat_member_restricted.ChatM	
attribute), 104	(aiogram.types.forum_topic_reopened.ForumTopicReopened
model_computed_fields	attribute), 134
	embinolipalatedinputed_fields (aiogram.types.game.Game
attribute), 116	attribute), 301
model_computed_fields	model_computed_fields
(aiogram.types.chat_permissions.ChatPermissions)	
attribute), 125	attribute), 302
model_computed_fields	model_computed_fields
(aiogram.types.chat_photo.ChatPhoto at- tribute), 126	<ul> <li>(aiogram.types.general_forum_topic_hidden.GeneralForumTopic attribute), 134</li> </ul>
model_computed_fields	model_computed_fields
(aiogram.types.chat_shared.ChatShared	(aiogram.types.general_forum_topic_unhidden.GeneralForumTopic_unhidden.Gene
attribute), 126	attribute), 134
model_computed_fields	model_computed_fields
(aiogram.types.chosen_inline_result.ChosenInl	
attribute), 229	135
model_computed_fields	model_computed_fields
(aiogram.types.contact.Contact attribute), 127	, (aiogram.types.giveaway_completed.GiveawayCompleted attribute), 135
<pre>model_computed_fields (aiogram.types.dice.Dice at-</pre>	- model_computed_fields
tribute), 127	(aiogram.types.giveaway_created.GiveawayCreated
model_computed_fields	attribute), 136
(aiogram.types.document.Document attribute),	
128	(aiogram.types.giveaway_winners.GiveawayWinners
model_computed_fields	attribute), 136
(aiogram.types.encrypted_credentials.Encrypte	
attribute), 281	(aiogram.types.inaccessible_message.InaccessibleMessage
model_computed_fields	attribute), 137
(aiogram.types.encrypted_passport_element.Enattribute), 282	(aiogram.types.inline_keyboard_button.InlineKeyboardButton
model_computed_fields	attribute), 138
(aiogram.types.error_event.ErrorEvent at-	
tribute), 553	(aiogram.types.inline_keyboard_markup.InlineKeyboardMarkup
model_computed_fields	attribute), 139
(aiogram.types.external_reply_info.ExternalRep	
attribute), 130	(aiogram.types.inline_query.InlineQuery
model_computed_fields (aiogram.types.file.File at-	
<i>tribute</i> ), 131	model_computed_fields
model_computed_fields	(aiogram.types.inline_query_result.InlineQueryResult
(aiogram.types.force_reply.ForceReply at-	
tribute), 132	model_computed_fields
model_computed_fields	(aiogram.types.inline_query_result_article.InlineQueryResultArt
(aiogram.types.forum_topic.ForumTopic	attribute), 233
attribute), 132	model_computed_fields
model_computed_fields	(aiogram.types.inline_query_result_audio.InlineQueryResultAud

attribute), 234

 $(a iogram.types.forum\_topic\_closed.ForumTopicClosed$ 

attribute), 269

```
model_computed_fields
                                                       model_computed_fields
         (aiogram.types.inline_query_result_cached_audio.InlineQuenyResult.CapdreitAindioquery_results_button.InlineQueryResultsB
                                                                attribute), 270
         attribute), 237
model_computed_fields
                                                       model_computed_fields
         (aiogram.types.inline_query_result_cached_document.Inline@iograReswpt@ainped_Doorumen_message_content.InputContactMes
                                                                attribute), 271
        attribute), 239
model_computed_fields
                                                       model_computed_fields
         (aiogram.types.inline_query_result_cached_gif.InlineQueryRainghtandypet&ifput_invoice_message_content.InputInvoiceMess
         attribute), 241
                                                                attribute), 273
model_computed_fields
                                                       model_computed_fields
         (aiogram.types.inline_query_result_cached_mpeg4_gif.Inline@ugrxyRexplaCiached_Mpatj&Gifnessage_content.InputLocationM
         attribute), 243
                                                                attribute), 274
model_computed_fields
                                                       model_computed_fields
         (aiogram.types.inline_query_result_cached_photo.InlineQuenippesult.CypeheilPhotomedia.InputMedia
         attribute), 245
                                                                attribute), 140
model_computed_fields
                                                       model_computed_fields
         (aiogram.types.inline_query_result_cached_sticker.InlineQuerivResultCpek.edfsitcheredia_animation.InputMediaAnimation
         attribute), 247
                                                                attribute), 141
model_computed_fields
                                                       model_computed_fields
         (aiogram.types.inline query result cached video.InlineQueryResaultGypekedWideomedia audio.InputMediaAudio
         attribute), 249
                                                                attribute), 142
model_computed_fields
                                                       model_computed_fields
         (aiogram.types.inline_query_result_cached_voice.InlineQuetxiRgsalttQsphsdNpitetemedia_document.InputMediaDocument
         attribute), 251
                                                                attribute), 143
model_computed_fields
                                                       model_computed_fields
         (aiogram.types.inline_query_result_contact.InlineQueryRes(di6gmtmctypes.input_media_photo.InputMediaPhoto
         attribute), 252
                                                                attribute), 144
model_computed_fields
                                                       model_computed_fields
         (aiogram.types.inline_query_result_document.InlineQueryRasiolgParattypest.input_media_video.InputMediaVideo
         attribute), 255
                                                                attribute), 145
                                                       model_computed_fields
model_computed_fields
         (aiogram.types.inline_query_result_game.InlineQueryResult@iongwam.types.input_message_content.InputMessageContent
        attribute), 256
                                                                attribute), 275
model_computed_fields
                                                       model_computed_fields
         (aiogram.types.inline_query_result_gif.InlineQueryResultGfaiogram.types.input_poll_option.InputPollOption
        attribute), 258
                                                                attribute), 146
model_computed_fields
                                                       model_computed_fields
         (aiogram.types.inline_query_result_location.InlineQueryRefalls@antitypes.input_sticker.InputSticker
         attribute), 260
                                                                attribute), 278
model_computed_fields
                                                       model_computed_fields
         (aiogram.types.inline_query_result_mpeg4_gif.InlineQueryReindtMpay#Esifnput_text_message_content.InputTextMessageCo
         attribute), 262
                                                                attribute), 276
model_computed_fields
                                                       model_computed_fields
         (aiogram.types.inline_query_result_photo.InlineQueryResultathogram.types.input_venue_message_content.InputVenueMessag
         attribute), 264
                                                                attribute), 276
model_computed_fields
                                                       model_computed_fields
         (aiogram.types.inline_query_result_venue.InlineQueryResultMengram.types.invoice.Invoice
                                                                                                  attribute),
        attribute), 266
model_computed_fields
                                                       model_computed_fields
         (aiogram.types.inline_query_result_video.InlineQueryResultWidgram.types.keyboard_button.KeyboardButton
         attribute), 268
                                                                attribute), 147
                                                       model_computed_fields
model_computed_fields
         (aiogram.types.inline_query_result_voice.InlineQueryResult\u00ediogram.types.keyboard_button_poll_type.KeyboardButtonPollT
```

676 Index

attribute), 147

model_computed_fields	model_computed_fields
	eyboardBut <b>(airRganessGhs</b> tmessage_origin_channel.MessageOriginChannel
attribute), 149	attribute), 207
model_computed_fields	model_computed_fields
	eyboardBut(airRequestypesmessage_origin_chat.MessageOriginChat
attribute), 150	attribute), 208
model_computed_fields	model_computed_fields
(aiogram.types.keyboard_button_request_users.k	KeyboardBu <mark>(taoonReapmexptesan</mark> essage_origin_hidden_user.MessageOriginHida
attribute), 151	attribute), 208
model_computed_fields	model_computed_fields
(aiogram.types.labeled_price.LabeledPrice attribute), 293	(aiogram.types.message_origin_user.MessageOriginUser attribute), 209
model_computed_fields	model_computed_fields
	wOptions (aiogram.types.message_reaction_count_updated.MessageReacti
attribute), 151	attribute), 209
model_computed_fields	model_computed_fields
(aiogram.types.location.Location attribute),	$(a iogram. types. message\_reaction\_updated. Message Reaction Updated. Message Reaction Updated$
152	attribute), 210
model_computed_fields	model_computed_fields
(aiogram.types.login_url.LoginUrl attribute), 153	(aiogram.types.order_info.OrderInfo attribute), 293
model_computed_fields	model_computed_fields
(aiogram.types.mask_position.MaskPosition attribute), 278	(aiogram.types.passport_data.PassportData attribute), 283
model_computed_fields	model_computed_fields
(aiogram.types.maybe_inaccessible_message.Maattribute), 153	ybeInacces(xibbe)Munstypes.passport_element_error.PassportElementError attribute), 284
model_computed_fields	model_computed_fields
(aiogram.types.menu_button.MenuButton attribute), 154	(aiogram.types.passport_element_error_data_field.PassportElemattribute), 284
model_computed_fields	model_computed_fields
	uttonComm( <b>nidg</b> ram.types.passport_element_error_file.PassportElementErr
attribute), 154	attribute), 285
model_computed_fields	model_computed_fields
(aiogram.types.menu_button_default.MenuButton	nDefault (aiogram.types.passport_element_error_files.PassportElementEr
attribute), 155	attribute), 286
model_computed_fields	model_computed_fields
	tonWebApp(aiogram.types.passport_element_error_front_side.PassportElem
attribute), 155	attribute), 287
model_computed_fields	model_computed_fields
(aiogram.types.message.Message attribute), 162	(aiogram.types.passport_element_error_reverse_side.PassportEleattribute), 288
model_computed_fields	model_computed_fields
	nged.Messa <b>gaAngnaDre.lspeAsi.pnas.Ghoan.geld</b> ement_error_selfie.PassportElementE
attribute), 205	attribute), 289
model_computed_fields	model_computed_fields
(aiogram.types.message_entity.MessageEntity attribute), 206	(aiogram.types.passport_element_error_translation_file.Passport attribute), 290
model_computed_fields	model_computed_fields
(aiogram types message id MessageId at-	(aigoram types passport element error translation files Passpor

attribute), 290

attribute), 291

 $(a iogram. types. passport\_element\_error\_unspecified. PassportElement\_error\_unspecified. PassportElement\_error\_unspecif$ 

model\_computed\_fields

tribute), 206

attribute), 207

 $(a iogram. types. message\_origin. MessageOrigin$ 

model\_computed\_fields

model_computed_fields	(aiogram.types.shipping_option.ShippingOption
(aiogram.types.passport_file.PassportFile	attribute), 296
attribute), 292	model_computed_fields
model_computed_fields	(aiogram.types.shipping_query.ShippingQuery
(aiogram.types.photo_size.PhotoSize attribute), 211	attribute), 296 model_computed_fields (aiogram.types.sticker.Sticker
model_computed_fields (aiogram.types.poll.Poll at-	attribute), 279
tribute), 212	model_computed_fields
model_computed_fields	(aiogram.types.sticker_set.StickerSet attribute),
(aiogram.types.poll_answer.PollAnswer at-	281
tribute), 212	<pre>model_computed_fields (aiogram.types.story.Story</pre>
model_computed_fields	attribute), 219
(aiogram.types.poll_option.PollOption at-	model_computed_fields
tribute), 213	(aiogram.types.successful_payment.SuccessfulPayment
model_computed_fields	attribute), 297
(aiogram.types.pre_checkout_query.PreCheckou	
attribute), 294	(aiogram.types.switch_inline_query_chosen_chat.SwitchInlineQu
model_computed_fields	attribute), 220
(aiogram.types.proximity_alert_triggered.Proxin	•
attribute), 213	(aiogram.types.text_quote.TextQuote attribute),
model_computed_fields	220
(aiogram.types.reaction_count.ReactionCount	model_computed_fields
attribute), 214	(aiogram.types.update.Update attribute),
model_computed_fields	299
(aiogram.types.reaction_type.ReactionType	model_computed_fields (aiogram.types.user.User at-
attribute), 214	tribute), 221
model_computed_fields	model_computed_fields
(atogram.types.reaction_type_custom_emoji.Rea attribute), 214	actionTypeC( <b>astogrEmaji</b> pes.user_chat_boosts.UserChatBoosts attribute), 222
model_computed_fields	model_computed_fields
(aiogram.types.reaction_type_emoji.ReactionTyp	
attribute), 215	attribute), 222
model_computed_fields	model_computed_fields
	eyboardMar <b>\aip</b> gram.types.user_shared.UserShared
attribute), 216	attribute), 223
model_computed_fields	model_computed_fields
	yboardRem(wiogram.types.users_shared.UsersShared
attribute), 216	attribute), 223
model_computed_fields	<pre>model_computed_fields (aiogram.types.venue.Venue</pre>
(aiogram.types.reply_parameters.ReplyParamete	ers attribute), 224
attribute), 217	<pre>model_computed_fields (aiogram.types.video.Video</pre>
model_computed_fields	attribute), 225
(aiogram.types.response_parameters.ResponsePa	
attribute), 218	(aiogram.types.video_chat_ended.VideoChatEnded
model_computed_fields	attribute), 225
(aiogram.types.sent_web_app_message.SentWeb	
attribute), 277	(aiogram.types.video_chat_participants_invited.VideoChatPartic
model_computed_fields	attribute), 226
(aiogram.types.shared_user.SharedUser	model_computed_fields
attribute), 218	(aiogram.types.video_chat_scheduled.VideoChatScheduled
model_computed_fields	attribute), 226
(aiogram.types.shipping_address.ShippingAddre	
attribute), 295	(aiogram.types.video_chat_started.VideoChatStarted
model_computed_fields	attribute), 226

- aiogram Documentation, Release 3.7.0 model\_computed\_fields model\_post\_init() (aiogram.methods.ban\_chat\_sender\_chat.BanChatSe (aiogram.types.video\_note.VideoNote method), 325 model\_post\_init() tribute), 227 (aiogram.methods.close.Close model\_computed\_fields method), 326 (aiogram.types.voice.Voice attribute), 227 model\_post\_init() (aiogram.methods.close\_forum\_topic.CloseForumTo model\_computed\_fields method), 327 (aiogram.types.web\_app\_data.WebAppData model\_post\_init() (aiogram.methods.close\_general\_forum\_topic.Close attribute), 228 method), 328 model\_computed\_fields model\_post\_init() (aiogram.methods.copy\_message.CopyMessage (aiogram.types.web\_app\_info.WebAppInfo method), 330 attribute), 228 model\_post\_init() (aiogram.methods.copy\_messages.CopyMessages model\_computed\_fields method), 332 (aiogram.types.webhook\_info.WebhookInfo model\_post\_init() (aiogram.methods.create\_chat\_invite\_link.CreateChat\_invite\_link.Crea attribute), 300 *method*), 333 model\_computed\_fields model\_post\_init() (aiogram.methods.create\_forum\_topic.CreateForum) (aiogram.types.write\_access\_allowed.WriteAccessAllowed\_method), 334 attribute), 229 model\_post\_init() (aiogram.methods.create\_invoice\_link.CreateInvoice model\_computed\_fields method), 460 model\_post\_init() (aiogram.methods.create\_new\_sticker\_set.CreateNew (aiogram.utils.web\_app.WebAppChat *tribute*), 576 method), 304 model\_computed\_fields model\_post\_init() (aiogram.methods.decline\_chat\_join\_request.Decline (aiogram.utils.web\_app.WebAppInitData method), 335 attribute), 574 model\_post\_init() (aiogram.methods.delete\_chat\_photo.DeleteChatPho method), 336 model\_computed\_fields (aiogram.utils.web\_app.WebAppUser model\_post\_init() (aiogram.methods.delete\_chat\_sticker\_set.DeleteCha tribute), 575 method), 337 model\_config (aiogram.utils.web\_app.WebAppChat atmodel\_post\_init() (aiogram.methods.delete\_forum\_topic.DeleteForum] tribute), 576 method), 338 model\_config (aiogram.utils.web\_app.WebAppInitData model\_post\_init() (aiogram.methods.delete\_message.DeleteMessage attribute), 574 method), 435 model\_config (aiogram.utils.web\_app.WebAppUser atmodel\_post\_init() (aiogram.methods.delete\_messages.DeleteMessages tribute), 575 method), 436 model\_fields (aiogram.utils.web\_app.WebAppChat atmodel\_post\_init() (aiogram.methods.delete\_my\_commands.DeleteMyC tribute), 577 method), 340 model\_fields (aiogram.utils.web\_app.WebAppInitData model\_post\_init() (aiogram.methods.delete\_sticker\_from\_set.DeleteStic attribute), 574 method), 305 model\_fields (aiogram.utils.web\_app.WebAppUser atmodel\_post\_init() (aiogram.methods.delete\_sticker\_set.DeleteStickerSe tribute), 576 method), 306
  - model\_post\_init() (aiogram.methods.add\_sticker\_to\_sembdeSticker\_to() (aiogram.methods.delete\_webhook.DeleteWebhook method), 303 method), 465
- model\_post\_init() (aiogram.methods.answer\_callback\_quadry.Apporrer\_CathlbuChQniergram.methods.edit\_chat\_invite\_link.EditChatInv
- method), 321 method), 341 model\_post\_init() (aiogram.methods.answer\_inline\_quamodhrlsyporhtliriaQtuety)(aiogram.methods.edit\_forum\_topic.EditForumTopic
- method), 449 method), 342
- model\_post\_init() (aiogram.methods.answer\_pre\_check**motlejuepp.stn.sivniR**t@CladokoatQmaethods.edit\_general\_forum\_topic.EditGen *method*), 456 method), 343
- model\_post\_init() (aiogram.methods.answer\_shipping\_moderlApposerShipping\_Quegram.methods.edit\_message\_caption.EditMessage method), 457 method), 437
- model\_post\_init() (aiogram.methods.answer\_web\_app\_**modes!AppsstrWahAfp)Qmicrg**ram.methods.edit\_message\_live\_location.EditM method), 450method), 439
- model\_post\_init() (aiogram.methods.approve\_chat\_joinmodplespApproveClQt.JainReagnestethods.edit\_message\_media.EditMessage method), 322 method), 441
- model\_post\_init() (aiogram.methods.ban\_chat\_membernRadaOhapteMemberit() (aiogram.methods.edit\_message\_reply\_markup.Editl *method*), 324 method), 442

- model\_post\_init() (aiogram.methods.edit\_message\_text1fioldeMexposseTäxtit() (aiogram.methods.log\_out.LogOut method), 444 method), 365
- model\_post\_init() (aiogram.methods.export\_chat\_invitembide.ExpostChailtv(i)elaiolgram.methods.pin\_chat\_message.PinChatMessage.method), 344 method), 366
- model\_post\_init() (aiogram.methods.forward\_message.fmodedrdfdestsagnit() (aiogram.methods.promote\_chat\_member.PromoteComethod), 346 method), 369
- model\_post\_init() (aiogram.methods.forward\_messagesmodes\updbetssigeist() (aiogram.methods.reopen\_forum\_topic.ReopenForum\_method), 347 method), 370
- model\_post\_init() (aiogram.methods.get\_business\_commodel\_GodBusiness\_Commethods.reopen\_general\_forum\_topic.Reopen\_method), 348

  method), 371
- model\_post\_init() (aiogram.methods.get\_chat.GetChat model\_post\_init() (aiogram.methods.replace\_sticker\_in\_set.ReplaceSticked), 349
  method), 309
- model\_post\_init() (aiogram.methods.get\_chat\_administ**modeslGptGhatiAnhri(i)s(minogrs**um.methods.restrict\_chat\_member.RestrictChat\_method), 350 method), 372
- model\_post\_init() (aiogram.methods.get\_chat\_member.@et@latMosnbanit() (aiogram.methods.revoke\_chat\_invite\_link.RevokeChat\_method), 351 method), 374
- model\_post\_init() (aiogram.methods.get\_chat\_member\_model\_GroChaiMirub) (Giogram.methods.send\_animation.SendAnimation method), 352 method), 376
- model\_post\_init() (aiogram.methods.get\_custom\_emoji\_woid&drsp6stCristionvC)(ajiSqicknrmethods.send\_chat\_action.SendChatAction\_method), 307 method), 381
- model\_post\_init() (aiogram.methods.get\_forum\_topic\_imodelicpostGittiot(in)(kiipignlaconsbittheds.send\_dice.SendDice method), 355 method), 385
- model\_post\_init() (aiogram.methods.get\_game\_high\_scmodeCletGestreHhighScowniogram.methods.send\_document.SendDocument method), 452 method), 387
- model\_post\_init() (aiogram.methods.get\_my\_commandsm6deWyfostminds() (aiogram.methods.send\_invoice.SendInvoice method), 357 method), 462
- model\_post\_init() (aiogram.methods.get\_my\_default\_admoidrestrapos\_trightist@n(MixIQefaultnAdmidissrendb\_rRightisn.SendLocation method), 358 method), 389
- model\_post\_init() (aiogram.methods.get\_my\_descriptiom@deMyDostription() (aiogram.methods.send\_media\_group.SendMediaGram.method), 359

  method), 392
- model\_post\_init() (aiogram.methods.get\_my\_name.GetMyNeInepost\_init() (aiogram.methods.send\_message.SendMessage method), 360 method), 394
- model\_post\_init() (aiogram.methods.get\_my\_short\_descriptedn\_forstMySthirtDexairigrium.methods.send\_photo.SendPhoto method), 361 method), 396
- model\_post\_init() (aiogram.methods.get\_sticker\_set.Getbodkdr\_Spost\_init() (aiogram.methods.send\_poll.SendPoll method), 308 method), 399
- model\_post\_init() (aiogram.methods.get\_user\_chat\_boa**noxd&lt\_lipss\*ClixaBb6)**t{aiogram.methods.send\_venue.SendVenue method), 362 method), 402
- model\_post\_init() (aiogram.methods.get\_user\_profile\_phodslGpdsterPhotslQPlaitogram.methods.send\_video.SendVideo method), 362 method), 405
- model\_post\_init() (aiogram.methods.get\_webhook\_info.modWebposkIrifacit() (aiogram.methods.send\_video\_note.SendVideoNote method), 467 method), 407
- model\_post\_init() (aiogram.methods.hide\_general\_forumoderlicpfoiste\_Genietal)F(aiorgillarpimethods.send\_voice.SendVoice method), 363 method), 410

- model\_post\_init() (aiogram.methods.set\_chat\_descriptionodelCloreDesirripti() (aiogram.methods.unpin\_all\_chat\_messages.UnpinA method), 413 method), 430
- model\_post\_init() (aiogram.methods.set\_chat\_menu\_buntodeSet@bstMinitR()toniogram.methods.unpin\_all\_forum\_topic\_messages.

  method), 414

  method), 431
- model\_post\_init() (aiogram.methods.set\_chat\_permissionod&etChoosRerimisti()n.(aiogram.methods.unpin\_all\_general\_forum\_topic\_n method), 415 method), 432
- model\_post\_init() (aiogram.methods.set\_chat\_photo.Set@ddePhptost\_init() (aiogram.methods.unpin\_chat\_message.UnpinChatM. method), 416 method), 433
- model\_post\_init() (aiogram.methods.set\_chat\_sticker\_s**mosterChpostckinSa**t() (aiogram.methods.upload\_sticker\_file.UploadSticker method), 417 method), 320
- model\_post\_init() (aiogram.methods.set\_chat\_title.SetChadWillepost\_init() (aiogram.types.animation.Animation method), 418 method), 19
- model\_post\_init() (aiogram.methods.set\_custom\_emoji\_woidedr\_postthimbt(i)l.SetCustoin@mojiStpekeas6di&lAundlbaail method), 312 method), 20
- model\_post\_init() (aiogram.methods.set\_game\_score.SamGdmleSposet\_init() (aiogram.types.background\_fill.BackgroundFill method), 455 method), 20
- model\_post\_init() (aiogram.methods.set\_message\_reactionds&t/MosstagiRitadi) (maiogram.types.background\_fill\_freeform\_gradient.B method), 420 method), 21
- model\_post\_init() (aiogram.methods.set\_my\_commandsm&dMy@ostmaindst() (aiogram.types.background\_fill\_gradient.Background\_method), 421 method), 21
- model\_post\_init() (aiogram.methods.set\_my\_default\_admoidislrapos\_trightistXe).MsiDefaultt\phrsi.lbistkeptonRid\_lfisl\_solid.BackgroundFil method), 423 method), 22
- model\_post\_init() (aiogram.methods.set\_my\_descriptiomSetEMly\_Postription() (aiogram.types.background\_type.BackgroundType method), 424 method), 22
- metnoa), 424 metnoa), 22
  model\_post\_init() (aiogram.methods.set\_my\_name.SetM**nodenl**epost\_init() (aiogram.types.background\_type\_chat\_theme.Backgr
- method), 424 method), 22
  model\_post\_init() (aiogram.methods.set\_my\_short\_desamiptidn\_forthy\_ShrirtDesamiptian.types.background\_type\_fill.BackgroundTyp
  method), 426 method), 23
- model\_post\_init() (aiogram.methods.set\_passport\_data<u>merdels\_SetPusiprirtDu(aForgram</u>.types.background\_type\_pattern.Background\_method), 470

  method), 23
- model\_post\_init() (aiogram.methods.set\_sticker\_emoji\_hiotl&dsSpioker\_FrniojiLis(aiogram.types.background\_type\_wallpaper.Backgro method), 313

  method), 24

  model\_post\_init() (aiogram methods set\_sticker\_kenyyambaKrl\Spiolark\Finit\Cle(aiogram types birthdate Birthdate
- model\_post\_init() (aiogram.methods.set\_sticker\_keywomdscset\SipicksrtKirnivtsGs(aiogram.types.birthdate.Birthdate method), 314 method), 25
- model\_post\_init() (aiogram.methods.set\_sticker\_mask\_modeidn\_SocstickeriMaskRingitiam.types.bot\_command.BotCommand method), 315 method), 25
- method), 315
  model\_post\_init() (aiogram.methods.set\_sticker\_positionodie\_lsep\$st\$tjirkiarHOs(taiondn&et.types.bot\_command\_scope.BotCommandSo
- method), 316
  method), 26
  model\_post\_init() (aiogram.methods.set\_sticker\_set\_thumoldedil\_poststickaise(T/(uiokmaik.types.bot\_command\_scope\_all\_chat\_admin\_method), 317
  method), 26
- model\_post\_init() (aiogram.methods.set\_sticker\_set\_tithmsdestiphosset\_firste() (aiogram.types.bot\_command\_scope\_all\_group\_channethod), 319

  method), 27
- model\_post\_init() (aiogram.methods.set\_webhook.SetWmbleedk\_post\_init() (aiogram.types.bot\_command\_scope\_all\_private\_chae method), 468
- model\_post\_init() (aiogram.methods.stop\_message\_livemboleeliapoStopMeista@elaineelpaenutiques.bot\_command\_scope\_chat.BotComm method), 446 method), 27
- model\_post\_init() (aiogram.methods.stop\_poll.StopPollmodel\_post\_init() (aiogram.types.bot\_command\_scope\_chat\_administration), 447
  method), 28
- model\_post\_init() (aiogram.methods.unban\_chat\_meml**modelb\_pash**tat**Meinko**n(aiogram.types.bot\_command\_scope\_chat\_member.l method), 427 method), 29 model\_post\_init() (aiogram.methods.unban\_chat\_sendemodelt\_tobolscinGhats@ndaiogram.types.bot\_command\_scope\_default.BotCom
- method), 428 method), 29
  model\_post\_init() (aiogram.methods.unhide\_general\_formore\_loppiostnliirli:G@n@nidformuntTyppicbot\_description.BotDescription
- model\_post\_init() (aiogram.methods.unhide\_general\_formorde\_loppiostr/lividic@nethod), 429 method), 429

method), 52

method), 56

```
model_post_init() (aiogram.types.bot_name.BotName model_post_init() (aiogram.types.chat_member_banned.ChatMemberEmethod), 30 method), 102

model_post_init() (aiogram.types.bot_short_descriptionmBdtShoptGsts_ciriptt()) (aiogram.types.chat_member_left.ChatMemberLeft method), 30 method), 102

model_post_init() (aiogram.types.business_connection.Blosiels.sqCostneinion() (aiogram.types.chat_member_member.ChatMemberlemethod), 31 method), 103

model_post_init() (aiogram.types.business intro.Businessinessintro) post_init() (aiogram.types.chat member owner.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMemberOver.ChatMem
```

- method), 31

  method), 103

  model\_post\_init() (aiogram.types.business\_location.Business\_location.Business\_location.Business\_method), 31

  method), 103

  method), 105
- model\_post\_init() (aiogram.types.business\_messages\_dwodel\_BpsixtesiMixs@gesDebatedtypes.chat\_member\_updated.ChatMember@method), 32

  method), 117
- model\_post\_init() (aiogram.types.business\_opening\_homodBilsipossQpiminffl(aiogram.types.chat\_permissions.ChatPermissions method), 32 method), 125
- model\_post\_init() (aiogram.types.business\_opening\_homodeilteposttBixiints(s)Oppingrghlappelnthatalphoto.ChatPhoto method), 33 method), 126
- model\_post\_init() (aiogram.types.callback\_query.Callback@lerpost\_init() (aiogram.types.chosen\_inline\_result.ChosenInlineRe method), 34
  method), 34
- model\_post\_init() (aiogram.types.chat.Chat method), model\_post\_init() (aiogram.types.contact.Contact method), 127
- model\_post\_init() (aiogram.types.chat\_administrator\_rightelleChpostdministratio(delleghtem.types.dice.Dice method), method), 51 127
- method), 52 method), 128 model\_post\_init() (aiogram.types.chat\_boost.ChatBoostmodel\_post\_init() (aiogram.types.encrypted\_credentials.EncryptedCredentials.En
- model\_post\_init() (aiogram.types.chat\_boost\_added.Chmcldedst\_dinit() (aiogram.types.encrypted\_passport\_element.Encrypted\_based\_init() (aiogram.types.encrypted\_based\_init() (aiogram.types.encrypted\_based\_

method), 281

method), 132

- model\_post\_init() (aiogram.types.chat\_boost\_removed.@bdrElogsofstrniviach() (aiogram.types.error\_event.ErrorEvent method), 53 method), 553
- model\_post\_init() (aiogram.types.chat\_boost\_source.ChanBedstSourceinit() (aiogram.types.external\_reply\_info.ExternalReplyInf method), 54 method), 130
- model\_post\_init() (aiogram.types.chat\_boost\_source\_giffo\_dedle\_pCost\_BinitSQur(aCifrandtypes.file.File method), method), 54
- model\_post\_init() (aiogram.types.chat\_boost\_source\_gimedædy\_pdsntBinitSQu(caiOginanntypes.force\_reply.ForceReply method), 55 method), 132
- model\_post\_init() (aiogram.types.chat\_boost\_source\_pmodeln\_postBionsis@n/aeiBgraniutypes.forum\_topic.ForumTopic method), 55 method), 132
- method), 55
  method), 132
  model\_post\_init() (aiogram.types.chat\_boost\_updated.@dodBbootdsptdated)t() (aiogram.types.forum\_topic\_closed.ForumTopicClos
- model\_post\_init() (aiogram.types.chat\_full\_info.ChatFut\( \text{bldfe}\) \_post\_init() (aiogram.types.forum\_topic\_created.ForumTopicCre
  method), 59
  method), 133
- model\_post\_init() (aiogram.types.chat\_join\_request.Ch**mcderlR\_papers**t\_init() (aiogram.types.forum\_topic\_reopened.ForumTopicRemethod), 93 method), 134
- model\_post\_init() (aiogram.types.chat\_location.ChatLomotleh\_post\_init() (aiogram.types.game.Game method), 99 method), 301 (aiogram.types.game.Game
- model\_post\_init() (aiogram.types.chat\_member\_admini**modtelr\_GlossM\_innbtr()dminigrant**sypes.general\_forum\_topic\_hidden.Genera method), 101 method), 134

- model\_post\_init() (aiogram.types.general\_forum\_topic\_modeildepo&enimilf@numTegriobltypidslenline\_query\_result\_venue.InlineQue method), 134 method), 266
- model\_post\_init() (aiogram.types.giveaway.Giveaway model\_post\_init() (aiogram.types.inline\_query\_result\_video.InlineQuententod), 135
  method), 268
- model\_post\_init() (aiogram.types.giveaway\_completed.@odelwap@stmphried() (aiogram.types.inline\_query\_result\_voice.InlineQuery\_method), 135 method), 269
- model\_post\_init() (aiogram.types.giveaway\_created.Giveadely\_Crostedinit() (aiogram.types.inline\_query\_results\_button.InlineQuerthod), 136 method), 270
- model\_post\_init() (aiogram.types.giveaway\_winners.Gimadely\_\vosterinit() (aiogram.types.input\_contact\_message\_content.Inpu method), 136 method), 271
- model\_post\_init() (aiogram.types.inaccessible\_messagewlordede\_spridste\_Massia\_6) (aiogram.types.input\_invoice\_message\_content.Inpu method), 137 method), 273 model\_post\_init() (aiogram.types.inline\_keyboard\_buttomodrelingKestb\_cinxilB(1)toniogram.types.input\_location\_message\_content.Inpu
- method), 138

  method), 274

  model post init() (aiogram.types.inline keyboard mar**knoolei**lli**nesks**tybionirilMoolkiapram.types.input media.InputMedia
- model\_post\_init() (aiogram.types.inline\_keyboard\_markqql&llippd&cybinairdMa(kiqgram.types.input\_media.InputMedia method), 139 method), 140
- method), 139

  method), 140

  model\_post\_init() (aiogram.types.inline\_query.InlineQuery.Inli
- model\_post\_init() (aiogram.types.inline\_query\_result.ImlockQuppyResihtit() (aiogram.types.input\_media\_audio.InputMediaAudio\_method), 232 method), 142
- model\_post\_init() (aiogram.types.inline\_query\_result\_amoidlell.ploseQianyRe)u(laArgiche.types.input\_media\_document.InputMediaL method), 233 method), 143
- model\_post\_init() (aiogram.types.inline\_query\_result\_amolde\_In\_liposQuaryiRe()ultahudinam.types.input\_media\_photo.InputMediaPhoto method), 234 method), 144
- model\_post\_init() (aiogram.types.inline\_query\_result\_cmothed\_apast\_lintiintQuentyDexultCypubeitApudionedia\_video.InputMediaVideo method), 237 method), 145
- model\_post\_init() (aiogram.types.inline\_query\_result\_cmolled\_dpostm\_intilt(in) (QueryResylveGanfied\_Docssaget\_content.InputMessage method), 239 method), 275 model\_post\_init() (aiogram.types.inline\_query\_result\_cmolled\_gifstlipteQueryNeisylveGanfied\_poll\_option.InputPollOption
- method), 241

  method), 146

  method), 146
- model\_post\_init() (aiogram.types.inline\_query\_result\_cmodel\_ppsg4\_igif\_t)()r(eQuerxyRexypltCimqhaddMjrkg4.Gif)utSticker method), 243 method), 278
- method), 243 method), 278
  model\_post\_init() (aiogram.types.inline\_query\_result\_c**molled\_plost\_lidiveQuenyBexult.CypelseilPho\_to**ext\_message\_content.InputTex
- model\_post\_init() (aiogram.types.inline\_query\_result\_cmolled\_spicktr\_IndirteQueningResult@peh.idftit\_keenue\_message\_content.InputVenthod), 247

  method), 277

method), 276

model\_post\_init() (aiogram.types.inline\_query\_result\_c**molled\_ridex\_LihinaQ)**very**RasidgCanhaypVicheo**voice.Invoice method), 249 method), 292

method), 245

- model\_post\_init() (aiogram.types.inline\_query\_result\_cmothed\_poist\_fribitieQ))drojRgsultaCophedVybaard\_button.KeyboardButton method), 251 method), 147
- model\_post\_init() (aiogram.types.inline\_query\_result\_cmodert\_lptist\_Qimit\_R)s(dif\_gmtanctypes.keyboard\_button\_poll\_type.Keyboard\_method), 253 method), 147
- model\_post\_init() (aiogram.types.inline\_query\_result\_dmodelempdshine@intx()RaxiolgPoontypestkeyboard\_button\_request\_chat.Keyboard, 149
- model\_post\_init() (aiogram.types.inline\_query\_result\_gmodeln|pasQueinyRes()|(Gianguam.types.keyboard\_button\_request\_user.Keyboard), 256 method), 150
- model\_post\_init() (aiogram.types.inline\_query\_result\_gij6HdIn@@rseryReixtl(Gffaiogram.types.keyboard\_button\_request\_users.Keyb
  method), 258

  method), 151
- model\_post\_init() (aiogram.types.inline\_query\_result\_lmcdteln\_pdsteQnartyResultgoantitypes.labeled\_price.LabeledPrice method), 260 method), 293
- model\_post\_init() (aiogram.types.inline\_query\_result\_mpoded\_gifbhtlirieQute(r) ReindtMpayfesifink\_preview\_options.LinkPreviewOpmethod), 262 method), 151
- model\_post\_init() (aiogram.types.inline\_query\_result\_phode\_InfposeQuaryiReQuit\_Phiotogram.types.location.Location method), 264 method), 152

- model\_post\_init() (aiogram.types.login\_url.LoginUrl model\_post\_init() (aiogram.types.passport\_element\_error\_translation\_method), 153 method), 290
- model\_post\_init() (aiogram.types.mask\_position.MaskPmoditeh\_post\_init() (aiogram.types.passport\_element\_error\_translation\_method), 278 method), 290
- model\_post\_init() (aiogram.types.maybe\_inaccessible\_modelpost\_bilmix()s(xibbeMunstyges.passport\_element\_error\_unspecified. method), 153 method), 291
- model\_post\_init() (aiogram.types.menu\_button\_comma**moddMerpdBstttcirtCictún(aiog**ram.types.photo\_size.PhotoSize method), 154 method), 211
- model\_post\_init() (aiogram.types.menu\_button\_default.**ModelBprostDafait**t() (aiogram.types.poll.Poll method), method), 155
- model\_post\_init() (aiogram.types.menu\_button\_web\_ap**modlehupostonWebApp**(aiogram.types.poll\_answer.PollAnswer method), 155 method), 212
- model\_post\_init() (aiogram.types.message\_auto\_delete\_mtodel\_\_aproxygedxMtx.SugarhagnaDeleypedispner\_Ghankodt\_query.PreCheckoutQue method), 205

  method), 294
- model\_post\_init() (aiogram.types.message\_entity.MessagedEnlitpost\_init() (aiogram.types.proximity\_alert\_triggered.ProximityAmethod), 206 method), 213
- model\_post\_init() (aiogram.types.message\_origin.Messmodeligipost\_init() (aiogram.types.reaction\_type.ReactionType method), 207 method), 214
- method), 207 method), 214
  model\_post\_init() (aiogram.types.message\_origin\_chamnoble/lespoge/Origint(I)duniogram.types.reaction\_type\_custom\_emoji.Reactio
- method), 207
  method), 214
  model\_post\_init() (aiogram.types.message\_origin\_chat.ModsalugpOsig\_initial() (aiogram.types.reaction\_type\_emoji.ReactionTypeEn
  method), 208
  method), 215
- model\_post\_init() (aiogram.types.message\_origin\_hiddmodeskrpbesstajenOtiQ)r(Hindelanhltyepes.reply\_keyboard\_markup.ReplyKeyboard), 208

  method), 216
- model\_post\_init() (aiogram.types.message\_origin\_user.**Mods**@gpOsigiriblitat() (aiogram.types.reply\_keyboard\_remove.ReplyKeyboard\_nethod), 209

  method), 216
- model\_post\_init() (aiogram.types.message\_reaction\_comod\_mpdptest\texsitge\t
- model\_post\_init() (aiogram.types.message\_reaction\_up**rlothellMpost**ge**Rkivt(i)r(lipdgmand**.types.response\_parameters.ResponseParameters.desponseParameters), 218

model\_post\_init() (aiogram.types.order\_info.OrderInfo model\_post\_init() (aiogram.types.sent\_web\_app\_message.SentWebApp

- method), 293 method), 277
  model\_post\_init() (aiogram.types.passport\_data.Passpamodaelu\_post\_init() (aiogram.types.shared\_user.SharedUser
- model\_post\_init() (aiogram.types.passport\_data.PasspomtdetL\_post\_init() (aiogram.types.shared\_user.SharedUser method), 283 method), 218
- model\_post\_init() (aiogram.types.passport\_element\_ermodelspposEleimirttE) (aiogram.types.shipping\_address.ShippingAddress method), 284 method), 295
- model\_post\_init() (aiogram.types.passport\_element\_ermodeldtap@std\_Paisp@rtEitergeanEtryperDdtipfpielg\_option.ShippingOption method), 285 method), 296
- model\_post\_init() (aiogram.types.passport\_element\_ermodfelk\_**Prosspointilty()** fullingontriltypes.shipping\_query.ShippingQuery method), 285 method), 296
- model\_post\_init() (aiogram.types.passport\_element\_ermod@lessPostportEtanentEr(aiventEr(aiventanent), 286 method), 279
- model\_post\_init() (aiogram.types.passport\_element\_ermodfebn\_poste\_PasspOrtEleguantEnyposFstinkSid\_eset.StickerSet method), 287 method), 281
- model\_post\_init() (aiogram.types.passport\_element\_ermod\_melveposs\_kidenRitsSportElementEgranReyverseSides.Story method), 288 method), 219
- model\_post\_init() (aiogram.types.passport\_element\_ermoodeelfipePertspioriElementSyffes.successful\_payment.SuccessfulPayment.od), 289

  method), 297

```
model_post_init() (aiogram.types.switch_inline_query_choseoi_orbanasiwendundineQueryChoseoi_Chat
                                                                                      aiogram.enums.dice_emoji, 478
             method), 220
                                                                                      aiogram.enums.encrypted_passport_element,
model_post_init() (aiogram.types.text quote.TextQuote
            method), 220
model_post_init()
                                   (aiogram.types.update.Update
                                                                                      aiogram.enums.inline_query_result_type,
            method), 299
                                                                                      aiogram.enums.input_media_type, 480
model_post_init() (aiogram.types.user.User method),
                                                                                      aiogram.enums.keyboard_button_poll_type_type,
model_post_init() (aiogram.types.user chat boosts.UserChatBoosts0
             method), 222
                                                                                      aiogram.enums.mask_position_point, 480
model_post_init() (aiogram.types.user_profile_photos.UserProfilePlant@nums.menu_button_type, 481
                                                                                      aiogram.enums.message_entity_type, 481
             method), 222
model_post_init() (aiogram.types.user_shared.UserShared
                                                                                      aiogram.enums.message_origin_type, 482
             method), 223
                                                                                      aiogram.enums.parse_mode, 482
model_post_init() (aiogram.types.users_shared.UsersSharedaiogram.enums.passport_element_error_type,
             method), 223
model_post_init()
                                                                                      aiogram.enums.poll_type, 483
                                       (aiogram.types.venue.Venue
            method), 224
                                                                                      aiogram.enums.reaction_type_type, 483
model_post_init()
                                                                                      aiogram.enums.sticker_format, 483
                                       (aiogram.types.video.Video
             method), 225
                                                                                      aiogram.enums.sticker_type, 483
model_post_init() (aiogram.types.video_chat_ended.VideoClacitbiogram.topic_icon_color, 484
            method), 225
                                                                                      aiogram.enums.update_type, 484
model_post_init() (aiogram.types.video_chat_participants_invitegh\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\dinlenkdepk\
                                                                                      aiogram.handlers.callback_query, 558
             method), 226
model_post_init()(aiogram.types.video_chat_scheduled.VidexiClgutSmhm\texthdods.add_sticker_to_set, 302
            method), 226
                                                                                      aiogram.methods.answer_callback_query,
model_post_init() (aiogram.types.video_chat_started.VideoChatStarted
                                                                                      aiogram.methods.answer_inline_query, 448
             method), 226
model_post_init() (aiogram.types.video_note.VideoNote
                                                                                      aiogram.methods.answer_pre_checkout_query,
             method), 227
model_post_init()
                                        (aiogram.types.voice.Voice
                                                                                      aiogram.methods.answer_shipping_query,
            method), 227
                                                                                             457
model_post_init() (aiogram.types.web_app_data.WebAppDataiogram.methods.answer_web_app_query, 450
                                                                                      aiogram.methods.approve_chat_join_request,
             method), 228
model_post_init() (aiogram.types.web_app_info.WebAppInfo
                                                                                      aiogram.methods.ban_chat_member, 323
            method), 228
model_post_init() (aiogram.types.webhook info.WebhookInfaiogram.methods.ban_chat_sender_chat, 325
             method), 300
                                                                                      aiogram.methods.close, 326
model_post_init() (aiogram.types.write_access_allowed.WritaAlogaxxxxllhoedtabds.close_forum_topic, 327
                                                                                      aiogram.methods.close_general_forum_topic,
            method), 229
model_post_init() (aiogram.utils.web app.WebAppChat
             method), 577
                                                                                      aiogram.methods.copy_message, 329
model_post_init() (aiogram.utils.web_app.WebAppInitData aiogram.methods.copy_messages, 331
             method), 574
                                                                                      aiogram.methods.create_chat_invite_link,
model_post_init() (aiogram.utils.web_app.WebAppUser
             method), 576
                                                                                      aiogram.methods.create_forum_topic, 334
module
                                                                                      aiogram.methods.create_invoice_link, 458
      aiogram.dispatcher.flags, 556
                                                                                      aiogram.methods.create_new_sticker_set,
      aiogram.enums.bot_command_scope_type, 471
                                                                                      aiogram.methods.decline_chat_join_request,
      aiogram.enums.chat_action, 471
      aiogram.enums.chat_boost_source_type, 472
                                                                                      aiogram.methods.delete_chat_photo, 336
      aiogram.enums.chat_member_status, 472
      aiogram.enums.chat_type, 473
                                                                                      aiogram.methods.delete_chat_sticker_set,
      aiogram.enums.content_type, 473
                                                                                            337
```

```
aiogram.methods.delete_forum_topic, 338
                                               aiogram.methods.hide_general_forum_topic,
                                                   363
aiogram.methods.delete_message, 434
aiogram.methods.delete_messages, 436
                                               aiogram.methods.leave_chat, 364
aiogram.methods.delete_my_commands, 339
                                               aiogram.methods.log_out, 365
aiogram.methods.delete_sticker_from_set,
                                               aiogram.methods.pin_chat_message, 366
                                               aiogram.methods.promote_chat_member, 367
aiogram.methods.delete_sticker_set, 306
                                               aiogram.methods.reopen_forum_topic, 370
aiogram.methods.delete_webhook, 464
                                               aiogram.methods.reopen_general_forum_topic,
aiogram.methods.edit_chat_invite_link,
                                               aiogram.methods.replace_sticker_in_set,
aiogram.methods.edit_forum_topic, 342
aiogram.methods.edit_general_forum_topic,
                                               aiogram.methods.restrict_chat_member, 372
                                               aiogram.methods.revoke_chat_invite_link,
aiogram.methods.edit_message_caption, 437
                                                   373
aiogram.methods.edit_message_live_location,
                                               aiogram.methods.send_animation, 375
    438
                                               aiogram.methods.send_audio, 377
                                               aiogram.methods.send_chat_action, 380
aiogram.methods.edit_message_media, 441
aiogram.methods.edit_message_reply_markup,
                                               aiogram.methods.send_contact, 382
                                               aiogram.methods.send_dice, 384
aiogram.methods.edit_message_text, 443
                                               aiogram.methods.send_document, 386
aiogram.methods.export_chat_invite_link,
                                               aiogram.methods.send_game, 452
                                               aiogram.methods.send_invoice, 461
                                               aiogram.methods.send_location, 388
aiogram.methods.forward_message, 345
aiogram.methods.forward_messages, 347
                                               aiogram.methods.send_media_group, 391
aiogram.methods.get_business_connection,
                                               aiogram.methods.send_message, 393
                                               aiogram.methods.send_photo, 395
aiogram.methods.get_chat, 349
                                               aiogram.methods.send_poll, 398
aiogram.methods.get_chat_administrators,
                                               aiogram.methods.send_sticker, 310
                                               aiogram.methods.send_venue, 401
aiogram.methods.get_chat_member, 351
                                               aiogram.methods.send_video, 403
aiogram.methods.get_chat_member_count,
                                               aiogram.methods.send_video_note, 406
    352
                                               aiogram.methods.send_voice, 409
aiogram.methods.get_chat_menu_button, 353
                                               aiogram.methods.set_chat_administrator_custom_title,
aiogram.methods.get_custom_emoji_stickers,
    307
                                               aiogram.methods.set_chat_description, 413
aiogram.methods.get_file, 354
                                               aiogram.methods.set_chat_menu_button, 414
aiogram.methods.get_forum_topic_icon_stickers,aiogram.methods.set_chat_permissions,415
                                               aiogram.methods.set_chat_photo, 416
aiogram.methods.get_game_high_scores, 451
                                               aiogram.methods.set_chat_sticker_set, 417
aiogram.methods.get_me, 356
                                               aiogram.methods.set_chat_title, 418
                                               aiogram.methods.set_custom_emoji_sticker_set_thumbnail
aiogram.methods.get_my_commands, 357
aiogram.methods.get_my_default_administrator_rights,2
                                               aiogram.methods.set_game_score, 454
aiogram.methods.get_my_description, 359
                                               aiogram.methods.set_message_reaction, 419
aiogram.methods.get_my_name, 360
                                               aiogram.methods.set_my_commands, 421
aiogram.methods.get_my_short_description,
                                               aiogram.methods.set_my_default_administrator_rights,
aiogram.methods.get_sticker_set, 308
                                               aiogram.methods.set_my_description, 423
aiogram.methods.get_updates, 465
                                               aiogram.methods.set_my_name, 424
aiogram.methods.get_user_chat_boosts, 361
                                               aiogram.methods.set_my_short_description,
aiogram.methods.get_user_profile_photos,
                                               aiogram.methods.set_passport_data_errors,
aiogram.methods.get_webhook_info, 467
                                                   469
```

```
aiogram.methods.set_sticker_emoji_list,
                                                   28
    313
                                               aiogram.types.bot_command_scope_chat_member,
aiogram.methods.set_sticker_keywords, 314
aiogram.methods.set_sticker_mask_position,
                                               aiogram.types.bot_command_scope_default,
aiogram.methods.set_sticker_position_in_set,
                                               aiogram.types.bot_description, 29
                                               aiogram.types.bot_name, 30
                                               aiogram.types.bot_short_description, 30
aiogram.methods.set_sticker_set_thumbnail,
                                               aiogram.types.business_connection, 30
aiogram.methods.set_sticker_set_title,
                                               aiogram.types.business_intro, 31
                                               aiogram.types.business_location, 31
aiogram.methods.set_webhook, 467
                                               aiogram.types.business_messages_deleted,
aiogram.methods.stop_message_live_location,
                                               aiogram.types.business_opening_hours, 32
   445
aiogram.methods.stop_poll, 447
                                               aiogram.types.business_opening_hours_interval,
aiogram.methods.unban_chat_member, 426
aiogram.methods.unban_chat_sender_chat,
                                               aiogram.types.callback_game, 301
                                               aiogram.types.callback_query, 33
aiogram.methods.unhide_general_forum_topic,
                                               aiogram.types.chat, 35
                                               aiogram.types.chat_administrator_rights,
aiogram.methods.unpin_all_chat_messages,
                                               aiogram.types.chat_background, 52
aiogram.methods.unpin_all_forum_topic_messagesaiogram.types.chat_boost,52
                                               aiogram.types.chat_boost_added, 53
aiogram.methods.unpin_all_general_forum_topic_amissgnamgnestypes.chat_boost_removed, 53
                                               aiogram.types.chat_boost_source, 54
                                               \verb"aiogram.types.chat_boost_source_gift_code",
aiogram.methods.unpin_chat_message, 433
aiogram.methods.upload_sticker_file, 320
aiogram.types.animation, 18
                                               aiogram.types.chat_boost_source_giveaway,
aiogram.types.audio, 19
aiogram.types.background_fill, 20
                                               aiogram.types.chat_boost_source_premium,
aiogram.types.background_fill_freeform_gradient, 55
                                               aiogram.types.chat_boost_updated, 56
aiogram.types.background_fill_gradient,
                                               aiogram.types.chat_full_info,56
                                               aiogram.types.chat_invite_link, 60
aiogram.types.background_fill_solid, 21
                                               aiogram.types.chat_join_request, 61
aiogram.types.background_type, 22
                                               aiogram.types.chat_location, 99
aiogram.types.background_type_chat_theme,
                                               aiogram.types.chat_member,99
                                               aiogram.types.chat_member_administrator,
aiogram.types.background_type_fill, 23
aiogram.types.background_type_pattern, 23
                                               aiogram.types.chat_member_banned, 102
aiogram.types.background_type_wallpaper,
                                               aiogram.types.chat_member_left, 102
                                               aiogram.types.chat_member_member, 103
                                               aiogram.types.chat_member_owner, 103
aiogram.types.birthdate, 25
                                               aiogram.types.chat_member_restricted, 104
aiogram.types.bot_command, 25
aiogram.types.bot_command_scope, 25
                                               aiogram.types.chat_member_updated, 105
aiogram.types.bot_command_scope_all_chat_adminaisugraamortypes.chat_permissions, 124
                                               aiogram.types.chat_photo, 125
aiogram.types.bot_command_scope_all_group_chatasiogram.types.chat_shared, 126
                                               aiogram.types.chosen_inline_result, 229
aiogram.types.bot_command_scope_all_private_chancesgram.types.contact, 127
                                               aiogram.types.dice, 127
aiogram.types.bot_command_scope_chat, 27
                                               aiogram.types.document, 128
aiogram.types.bot_command_scope_chat_administrantomorsam.types.encrypted_credentials, 281
```

```
aiogram.types.encrypted_passport_element,
                                               aiogram.types.inline_query_result_location,
    282
                                                   258
aiogram.types.error_event, 553
                                               aiogram.types.inline_query_result_mpeg4_gif,
aiogram.types.external_reply_info, 128
aiogram.types.file, 131
                                               aiogram.types.inline_query_result_photo,
aiogram.types.force_reply, 131
aiogram.types.forum_topic, 132
                                               aiogram.types.inline_query_result_venue,
aiogram.types.forum_topic_closed, 132
aiogram.types.forum_topic_created, 133
                                               aiogram.types.inline_query_result_video,
aiogram.types.forum_topic_edited, 133
aiogram.types.forum_topic_reopened, 134
                                               aiogram.types.inline_query_result_voice,
aiogram.types.game, 301
aiogram.types.game_high_score, 302
                                               aiogram.types.inline_query_results_button,
aiogram.types.general_forum_topic_hidden,
                                               aiogram.types.input_contact_message_content,
aiogram.types.general_forum_topic_unhidden,
                                                   271
    134
                                               aiogram.types.input_file, 139
aiogram.types.giveaway, 134
                                               aiogram.types.input_invoice_message_content,
aiogram.types.giveaway_completed, 135
aiogram.types.giveaway_created, 136
                                               aiogram.types.input_location_message_content,
aiogram.types.giveaway_winners, 136
                                                   274
aiogram.types.inaccessible_message, 137
                                               aiogram.types.input_media, 140
aiogram.types.inline_keyboard_button, 137
                                               aiogram.types.input_media_animation, 140
aiogram.types.inline_keyboard_markup, 139
                                               aiogram.types.input_media_audio, 141
aiogram.types.inline_query, 230
                                               aiogram.types.input_media_document, 142
aiogram.types.inline_query_result, 231
                                               aiogram.types.input_media_photo, 144
aiogram.types.inline_query_result_article,
                                               aiogram.types.input_media_video, 144
                                               aiogram.types.input_message_content, 275
aiogram.types.inline_query_result_audio,
                                               aiogram.types.input_poll_option, 146
                                               aiogram.types.input_sticker, 277
aiogram.types.inline_query_result_cached_audioaiogram.types.input_text_message_content,
                                                   275
aiogram.types.inline_query_result_cached_documeintgram.types.input_venue_message_content,
                                                   276
aiogram.types.inline_query_result_cached_gif, aiogram.types.invoice, 292
                                               aiogram.types.keyboard_button, 146
aiogram.types.inline_query_result_cached_mpeg4aigigfram.types.keyboard_button_poll_type,
aiogram.types.inline_query_result_cached_photoaiogram.types.keyboard_button_request_chat,
aiogram.types.inline_query_result_cached_stickeriogram.types.keyboard_button_request_user,
aiogram.types.inline_query_result_cached_videogiogram.types.keyboard_button_request_users,
aiogram.types.inline_query_result_cached_voiceaiogram.types.labeled_price, 293
                                               aiogram.types.link_preview_options, 151
aiogram.types.inline_query_result_contact,
                                               aiogram.types.location, 152
                                               aiogram.types.login_url, 153
aiogram.types.inline_query_result_document,
                                               aiogram.types.mask_position, 278
                                               aiogram.types.maybe_inaccessible_message,
                                                   153
aiogram.types.inline_query_result_game,
                                               aiogram.types.menu_button, 154
aiogram.types.inline_query_result_gif,
                                               aiogram.types.menu_button_commands, 154
    256
                                               aiogram.types.menu_button_default, 155
```

```
aiogram.types.menu_button_web_app, 155
                                                 aiogram.types.sent_web_app_message, 277
aiogram.types.message, 156
                                                 aiogram.types.shared_user, 218
aiogram.types.message_auto_delete_timer_changeatiogram.types.shipping_address, 295
                                                 aiogram.types.shipping_option, 295
aiogram.types.message_entity, 205
                                                 aiogram.types.shipping_query, 296
aiogram.types.message_id, 206
                                                 aiogram.types.sticker, 279
aiogram.types.message_origin, 207
                                                 aiogram.types.sticker_set, 280
aiogram.types.message_origin_channel, 207
                                                 aiogram.types.story, 219
aiogram.types.message_origin_chat, 208
                                                 aiogram.types.successful_payment, 297
aiogram.types.message_origin_hidden_user,
                                                 aiogram.types.switch_inline_query_chosen_chat,
aiogram.types.message_origin_user, 209
                                                 aiogram.types.text_quote, 220
aiogram.types.message_reaction_count_updated,
                                                 aiogram.types.update, 298
                                                 aiogram.types.user, 220
                                                 aiogram.types.user_chat_boosts, 222
aiogram.types.message_reaction_updated,
    210
                                                 aiogram.types.user_profile_photos, 222
                                                 aiogram.types.user_shared, 223
aiogram.types.order_info, 293
aiogram.types.passport_data, 283
                                                 aiogram.types.users_shared, 223
aiogram.types.passport_element_error, 283
                                                 aiogram.types.venue, 224
aiogram.types.passport_element_error_data_fielatiogram.types.video, 224
    284
                                                 aiogram.types.video_chat_ended, 225
aiogram.types.passport_element_error_file,
                                                 aiogram.types.video_chat_participants_invited,
                                                      226
                                                 aiogram.types.video_chat_scheduled, 226
aiogram.types.passport_element_error_files,
                                                 {\tt aiogram.types.video\_chat\_started, 226}
aiogram.types.passport_element_error_front_sideiogram.types.video_note, 227
                                                 aiogram.types.voice, 227
aiogram.types.passport_element_error_reverse_saidegram.types.web_app_data, 228
                                                 aiogram.types.web_app_info, 228
aiogram.types.passport_element_error_selfie,
                                                 aiogram.types.webhook_info, 300
                                                 aiogram.types.write_access_allowed, 229
aiogram.types.passport_element_error_transilantibmafiglem.types.birthdate.Birthdate attribute), 25
                                             MOUTH (aiogram.enums.mask_position_point.MaskPositionPoint
aiogram.types.passport_element_error_translation_fixhrelsute), 480
                                             mpeg4_duration(aiogram.types.inline_query_result_mpeg4_gif.InlineQu
aiogram.types.passport_element_error_unspecified, attribute), 262
    291
                                             mpeg4_file_id (aiogram.types.inline_query_result_cached_mpeg4_gif.In
aiogram.types.passport_file, 292
                                                      attribute), 243
aiogram.types.photo_size, 211
                                             MPEG4_GIF (aiogram.enums.inline_query_result_type.InlineQueryResultTy
aiogram.types.poll, 211
                                                      attribute), 479
aiogram.types.poll_answer, 212
                                             mpeg4_height (aiogram.types.inline_query_result_mpeg4_gif.InlineQuery
aiogram.types.poll_option, 213
                                                      attribute), 262
aiogram.types.pre_checkout_query, 294
                                             mpeg4_url (aiogram.types.inline_query_result_mpeg4_gif.InlineQueryRes
aiogram.types.proximity_alert_triggered,
                                                      attribute), 262
                                             mpeg4_width(aiogram.types.inline_query_result_mpeg4_gif.InlineQueryI
aiogram.types.reaction_count, 214
                                                      attribute), 262
aiogram.types.reaction_type, 214
                                             MUR (aiogram.enums.currency.Currency attribute), 476
aiogram.types.reaction_type_custom_emoji,
                                             MVR (aiogram.enums.currency.Currency attribute), 477
    214
                                             MXN (aiogram.enums.currency.Currency attribute), 477
aiogram.types.reaction_type_emoji, 215
                                             MY_CHAT_MEMBER (aiogram.enums.update_type.UpdateType
aiogram.types.reply_keyboard_markup,215
                                                      attribute), 485
aiogram.types.reply_keyboard_remove, 216
                                             my_chat_member
                                                              (aiogram.types.update.Update
aiogram.types.reply_parameters, 217
                                                      tribute), 299
aiogram.types.response_parameters, 218
                                             MYR (aiogram.enums.currency.Currency attribute), 477
```

```
MZN (aiogram.enums.currency.Currency attribute), 477
                                                                                                                                             need_phone_number (aiogram.methods.create_invoice_link.CreateInvoice
                                                                                                                                                                     attribute), 460
N
                                                                                                                                             need_phone_number (aiogram.methods.send invoice.SendInvoice
                                                                                                                                                                     attribute), 463
name (aiogram.methods.add_sticker_to_set.AddStickerToSet
                                                                                                                                             need_phone_number (aiogram.types.input_invoice_message_content.Inpu
                       attribute), 303
\verb|name|| (aiogram.methods.create\_chat\_invite\_link.CreateChatInviteLink | attribute), 273
                                                                                                                                             need_shipping_address
                       attribute), 333
                                                                                                                                                                     (aiogram.methods.create_invoice_link.CreateInvoiceLink
name (aiogram.methods.create_forum_topic.CreateForumTopic
                                                                                                                                                                     attribute), 460
                       attribute), 334
name(aiogram.methods.create_new_sticker_set.CreateNew_$\text{Rexd}_r$\text{e}\text{ipping_address}
                                                                                                                                                                     (aiogram.methods.send_invoice.SendInvoice
                        attribute), 304
                                                                                                                                                                     attribute), 463
name (aiogram.methods.delete_sticker_set.DeleteStickerSet
                                                                                                                                             need_shipping_address
                       attribute), 306
                                                                                                                                                                     (aiogram.types.input_invoice_message_content.InputInvoiceMess
name (aiogram.methods.edit_chat_invite_link.EditChatInviteLink
                                                                                                                                                                     attribute), 274
                       attribute), 341
\verb|name| (a iogram.methods.edit\_forum\_topic.EditForumTopic| | \verb|needs\_repainting| (a iogram.methods.create\_new\_sticker\_set.CreateNews.edit\_forum\_topic.EditForumTopic| | \verb|needs\_repainting| (a iogram.methods.create\_new\_sticker\_set.CreateNews.edit\_forum\_topic.EditForum\_topic.EditForumTopic| | \verb|needs\_repainting| (a iogram.methods.create\_new\_sticker\_set.CreateNews.edit\_forum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topic.EditForum\_topi
                                                                                                                                                                     attribute), 304
                       attribute), 342
name (aiogram.methods.edit_general_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGeneral_forum_topic.EditGen
                                                                                                                                                                                                               (aiogram.types.sticker.Sticker
                                                                                                                                                                     attribute), 280
                       attribute), 343
                                                                                                                                            new_chat_member(aiogram.types.chat member updated.ChatMemberUp.
name (aiogram.methods.get_sticker_set.GetStickerSet at-
                                                                                                                                                                     attribute), 106
                       tribute), 308
name (aiogram.methods.replace_sticker_in_set.ReplaceSticker_MSCHAT_MEMBERS (aiogram.enums.content_type.ContentType
                                                                                                                                                                     attribute), 474
                       attribute), 309
name (aiogram.methods.set_custom_emoji_sticker_set_thumbawiishatcusomberis)ishekessetttindasage.Message
                                                                                                                                                                     attribute), 160
                       attribute), 312
                                                                                                                                             NEW_CHAT_PHOTO (aiogram.enums.content_type.ContentType
                              (aiogram.methods.set_my_name.SetMyName
name
                                                                                                                                                                     attribute), 474
                       attribute), 424
name (aiogram.methods.set_sticker_set_thumbnail.SetStickersetThhabnahoto (aiogram.types.message.Message at-
                                                                                                                                                                     tribute), 160
                        attribute), 317
attribute), 474
                       attribute), 319
                                                                                                                                             new_chat_title (aiogram.types.message.Message at-
name (aiogram.types.bot_name.BotName attribute), 30
                                                                                                                                                                     tribute), 160
name (aiogram.types.chat invite link.ChatInviteLink at-
                                                                                                                                             new_reaction(aiogram.types.message_reaction_updated.MessageReaction)
                        tribute), 60
                                                                                                                                                                     attribute), 210
name (aiogram.types.forum_topic.ForumTopic attribute),
                                                                                                                                             next_offset(aiogram.methods.answer_inline_query.AnswerInlineQuery
                                                                                                                                                                     attribute), 449
{\tt name}\,(aiogram.types.forum\_topic\_created.ForumTopicCreated
                                                                                                                                             NGN (aiogram.enums.currency.Currency attribute), 477
                       attribute), 133
NOK (aiogram.enums.currency.Currency attribute), 477
                        attribute), 133
                                                                                                                                             NPR (aiogram.enums.currency.Currency attribute), 477
name (aiogram.types.order_info.OrderInfo attribute), 293
                                                                                                                                             NZD (aiogram.enums.currency.Currency attribute), 477
name (aiogram.types.sticker_set.StickerSet attribute), 280
need_email (aiogram.methods.create_invoice_link.CreateInvoiceLink
                        attribute), 460
need_email(aiogram.methods.send invoice.SendInvoice offset (aiogram.methods.get updates.GetUpdates at-
                       attribute), 463
                                                                                                                                                                     tribute), 465
need_email(aiogram.types.input_invoice_message_contentafficeth(voicementafficeth) (voicementafficeth) (voi
                       attribute), 273
                                                                                                                                                                     attribute), 362
need_name (aiogram.methods.create_invoice_link.CreateInvoiteEthuk (aiogram.types.inline_query.InlineQuery
                       attribute), 460
                                                                                                                                                                     tribute), 230
need_name (aiogram.methods.send_invoice.SendInvoice offset (aiogram.types.message_entity.MessageEntity at-
                       attribute), 463
                                                                                                                                                                     tribute), 206
need_name (aiogram.types.input_invoice_message_content.dbg/untilngxaine:Met/soudge.Gnswent_pre_checkout_query.AnswerPreCheckoutQue
                                                                                                                                                                     attribute), 456
                       attribute), 273
```

```
ok (aiogram.methods.answer_shipping_query.AnswerShippipaQsermode (aiogram.methods.send_photo.SendPhoto
                                                                                                                                                                                                                               attribute), 396
                               attribute), 457
old_chat_member (aiogram.types.chat_member_updated.@atskemberUpdated;ram.methods.send_video.SendVideo
                               attribute), 106
                                                                                                                                                                                                                               attribute), 405
old_reaction(aiogram.types.message_reaction_updated.Marsageanardetion(dipalphardetion).methods.send_voice.SendVoice
                               attribute), 210
                                                                                                                                                                                                                               attribute), 410
old_sticker (aiogram.methods.replace sticker in set.Repsacesticker@action.types.inline query result audio.InlineQueryResultA
                                                                                                                                                                                                                               attribute), 234
                                attribute), 309
one_time_keyboard (aiogram.types.reply_keyboard_markppnsep_lynKelekouioodMunrkupes.inline_query_result_cached_audio.InlineQuer
                               attribute), 216
                                                                                                                                                                                                                               attribute), 237
only\_if\_banned (a iogram.methods.unban\_chat\_member. \cite{LpdxsxClmodde} (nbegram.types.inline\_query\_result\_cached\_document. In line \cite{LpdxsxClmodde} (nbegram.types.inline \cite{LpdxsxClmodde} (nbegram.types.types.inline \cite{LpdxsxClmodde} (nbegram.
                               attribute), 427
                                                                                                                                                                                                                               attribute), 239
only_new_members (aiogram.types.giveaway.Giveaway parse_mode (aiogram.types.inline_query_result_cached_gif.InlineQueryRe
                               attribute), 135
                                                                                                                                                                                                                               attribute), 241
only_new_members (aiogram.types.giveaway_winners.Giveawaye\texts\)inoder.(aiogram.types.inline_query_result_cached_mpeg4_gif.Inline
                               attribute), 137
                                                                                                                                                                                                                               attribute), 243
open_period (aiogram.methods.send_poll.SendPoll at- parse_mode (aiogram.types.inline_query_result_cached_photo.InlineQuer
                               tribute), 399
                                                                                                                                                                                                                               attribute), 245
open_period (aiogram.types.poll.Poll attribute), 212
                                                                                                                                                                                               parse_mode (aiogram.types.inline_query_result_cached_video.InlineQuery_
opening_hours (aiogram.types.business_opening_hours.BusinessOpettinkulte)ur349
                               attribute), 32
                                                                                                                                                                                               parse_mode (aiogram.types.inline_query_result_cached_voice.InlineQuery_
opening_minute(aiogram.types.business_opening_hours_interval.Battinibust@p@fingHoursInterval
                               attribute), 33
                                                                                                                                                                                               parse_mode (aiogram.types.inline_query_result_document.InlineQueryRes
option_ids (aiogram.types.poll_answer.PollAnswer at-
                                                                                                                                                                                                                               attribute), 255
                               tribute), 212
                                                                                                                                                                                               parse_mode (aiogram.types.inline_query_result_gif.InlineQueryResultGif
options
                                         (aiogram.methods.send_poll.SendPoll
                                                                                                                                                                              at-
                                                                                                                                                                                                                               attribute), 258
                               tribute), 399
                                                                                                                                                                                               parse_mode (aiogram.types.inline_query_result_mpeg4_gif.InlineQueryRe
options (aiogram.types.poll.Poll attribute), 211
                                                                                                                                                                                                                               attribute), 262
order_info(aiogram.types.pre_checkout_query.PreCheckquatQuery.mode(aiogram.types.inline_query_result_photo.InlineQueryResult}
                               attribute), 294
                                                                                                                                                                                                                               attribute), 264
order_info(aiogram.types.successful_payment.Successful_payment.ode(aiogram.types.inline_query_result_video.InlineQueryResultV
                               attribute), 297
                                                                                                                                                                                                                               attribute), 268
                                                                                                                                                                                               \verb"parse_mode" (a iogram.types.in line_query_result_voice. In line Query Result Voice) and the parse_mode of the parse_
OrderInfo (class in aiogram.types.order_info), 293
\verb"origin" (a iogram.types.external\_reply\_info. ExternalReplyInfo
                                                                                                                                                                                                                               attribute), 269
                               attribute), 129
                                                                                                                                                                                               parse_mode (aiogram.types.input_media_animation.InputMediaAnimation
                                                                                                                                                                                                                               attribute), 141
P
                                                                                                                                                                                               parse_mode (aiogram.types.input_media_audio.InputMediaAudio
                                                                                                                                                                                                                               attribute), 142
PAB (aiogram.enums.currency.Currency attribute), 477
                                                                                                                                                                                              parse_mode (aiogram.types.input_media_document.InputMediaDocument
                                        (aiogram.filters.callback data.CallbackData
                                                                                                                                                                                                                               attribute), 143
                               method), 506
attribute), 144
                               attribute), 330
{\tt parse\_mode} \ (aiogram.methods.edit\_message\_caption. Edit {\tt NPESSEge} {\tt Code} \ (aiogram.types.input\_media\_video. Input Media Video, the control of t
                                                                                                                                                                                                                                attribute), 145
                               attribute), 437
\verb|parse_mode|| (a iogram.methods.edit\_message\_text.EditMessage\_text.EditMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.editMessage\_text.edit
                                                                                                                                                                                                                                attribute), 276
                               attribute), 444
\verb|parse_mode|| (a iogram.methods.send_animation.SendAnimapan* se_webapp_init_data()|
                                                                                                                                                                                                                                                                                                                             (in
                                                                                                                                                                                                                                                                                                                                                               module
                                                                                                                                                                                                                               aiogram.utils.web_app), 573
                               attribute), 376
                                                                                                                                                                                              ParseMode (class in aiogram.enums.parse_mode), 482
parse_mode (aiogram.methods.send_audio.SendAudio
                                                                                                                                                                                               {\tt PASSPORT}\ (aiogram.enums.encrypted\_passport\_element. Encrypted Passport\_element. Encrypted Passport\_element.
                               attribute), 378
                                                                                                                                                                                                                                attribute), 478
parse_mode(aiogram.methods.send_document.SendDocument
                                                                                                                                                                                               PASSPORT_DATA (aiogram.enums.content_type.ContentType
                               attribute), 387
                                                                                                                                                                                                                               attribute), 474
parse_mode (aiogram.methods.send_message.SendMessage
                                                                                                                                                                                               passport_data
                                                                                                                                                                                                                                                                            (aiogram.types.message.Message
                               attribute), 394
```

```
attribute), 161
                                                                 attribute), 61
PASSPORT_REGISTRATION
                                                       pending_update_count
         (aiogram.enums.encrypted passport element.EncryptedPas(pointElenterpes.webhook info.WebhookInfo
                                                                 attribute), 300
         attribute), 479
PassportData (class in aiogram.types.passport_data), performer (aiogram.methods.send_audio.SendAudio at-
                                                                 tribute), 379
PassportElementError
                                                   in performer (aiogram.types.audio.Audio attribute), 19
                                   (class
         aiogram.types.passport_element_error),
                                                       performer(aiogram.types.inline_query_result_audio.InlineQueryResultAt
                                                                 attribute), 235
         283
PassportElementErrorDataField
                                                   in performer(aiogram.types.input_media_audio.InputMediaAudio
                                        (class
         aiogram.types.passport_element_error_data_field),
                                                                 attribute), 142
         284
                                                       permissions (aiogram.methods.restrict_chat_member.RestrictChatMembe
PassportElementErrorFile
                                     (class
                                                   in
                                                                 attribute), 372
         aiogram.types.passport_element_error_file),
                                                       permissions (aiogram.methods.set_chat_permissions.SetChatPermissions
                                                                 attribute), 415
PassportElementErrorFiles
                                      (class
                                                       permissions (aiogram.types.chat.Chat attribute), 38
                                                       permissions(aiogram.types.chat_full_info.ChatFullInfo
         aiogram.types.passport_element_error_files),
                                                                 attribute), 59
PassportElementErrorFrontSide
                                                   in personal_chat (aiogram.types.chat.Chat attribute), 38
                                        (class
         aiogram.types.passport_element_error_front_sidepersonal_chat (aiogram.types.chat_full_info.ChatFullInfo
         287
                                                                 attribute), 58
PassportElementErrorReverseSide
                                         (class
                                                   in PERSONAL_DETAILS (aiogram.enums.encrypted_passport_element.Encrypt
         aiogram.types.passport_element_error_reverse_side),
                                                                 attribute), 478
                                                       PHONE_NUMBER (aiogram.enums.encrypted_passport_element.EncryptedPa
PassportElementErrorSelfie
                                                                 attribute), 479
                                      (class
                                                   in
         aiogram.types.passport_element_error_selfie),
                                                       PHONE_NUMBER (aiogram.enums.message_entity_type.MessageEntityType
                                                                 attribute), 481
PassportElementErrorTranslationFile (class in
                                                       phone_number (aiogram.methods.send_contact.SendContact
         aiogram.types.passport_element_error_translation_file),
                                                                attribute), 382
                                                       phone_number
                                                                         (aiogram.types.contact.Contact
PassportElementErrorTranslationFiles (class in
                                                                 tribute), 127
         aiogram.types.passport_element_error_translatioph@hae_number (aiogram.types.encrypted_passport_element.EncryptedPass
         290
                                                                 attribute), 282
PassportElementErrorType
                                                       phone_number (aiogram.types.inline_query_result_contact.InlineQueryRe
                                     (class
                                                   in
         aiogram.enums.passport_element_error_type),
                                                                 attribute), 252
                                                       phone_number (aiogram.types.input_contact_message_content.InputConta
PassportElementErrorUnspecified
                                         (class
                                                                 attribute), 271
         aiogram.types.passport_element_error_unspecifiepthone_number (aiogram.types.order_info.OrderInfo at-
                                                                 tribute), 293
PassportFile (class in aiogram.types.passport_file), PhoneNumber (class in aiogram.utils.formatting), 585
                                                                (aiogram.enums.content type.ContentType at-
{\tt pattern}\ (aiogram. filters. exception. Exception Message Filter)
                                                                 tribute), 473
         attribute), 509
                                                       PHOTO (aiogram.enums.inline_query_result_type.InlineQueryResultType
pay (aiogram.types.inline_keyboard_button.InlineKeyboardButton
                                                                 attribute), 479
         attribute), 138
                                                       PHOTO (aiogram.enums.input_media_type.InputMediaType
\verb"payload" (a iogram.methods.create_invoice\_link.CreateInvoiceLink")
                                                                attribute), 480
                                                                (aiogram.methods.send_photo.SendPhoto
         attribute), 459
                                                       photo
payload (aiogram.methods.send_invoice.SendInvoice at-
                                                                 tribute), 396
         tribute), 462
                                                       photo
                                                                (aiogram.methods.set_chat_photo.SetChatPhoto
payload (aiogram.types.input_invoice_message_content.InputInvoiceMessbageContent
         attribute), 273
                                                       photo (aiogram.types.chat.Chat attribute), 38
PEN (aiogram.enums.currency.Currency attribute), 477
                                                               (aiogram.types.chat_full_info.ChatFullInfo
                                                       photo
                                                                                                          at-
pending_join_request_count
                                                                 tribute), 58
         (aiogram.types.chat invite link.ChatInviteLink photo
                                                                 (aiogram.types.chat shared.ChatShared
                                                                                                          at-
```

<i>tribute</i> ), 126	
photo (aiogram.types.external_reply_info.ExternalReplyInf\(p\)inned_message (aiogram.types.external_reply_info.ExternalReplyInf\(p\)inned_message (aiogram.types.external_reply_info.ExternalReplyInf\(p\)inned_message (aiogram.types.external_reply_info.ExternalReplyInf\(p\)inned_message (aiogram.types.external_reply_info.ExternalReplyInf\(p\)inned_message (aiogram.types.external_reply_info.ExternalReplyInf\(p\)inned_message (aiogram.types.externalReplyInf\(p\)inned_message (aiogram.types.exter	types.chat_full_info.ChatFullInfo
photo (aiogram.types.game.Game attribute), 301 pinned_message (aiogram	types.message.Message at-
photo (aiogram.types.message.Message attribute), 159 tribute), 161	
photo (aiogram.types.shared_user.SharedUser at- PKR (aiogram.enums.currence	y.Currency attribute), 477
tribute), 219 PLN (aiogram.enums.currenc	
photo_file_id(aiogram.types.inline_query_result_cache painoto.lnling@metyyRessultsSk attribute), 245 tribute), 278	a <b>chesliRhatM</b> askPosition at-
photo_height (aiogram.methods.create_invoice_link.Crea <b>PGuV</b> oice(hinkgram.enums.con attribute), 460 tribute), 473	tent_type.ContentType at-
photo_height (aiogram.methods.send_invoice.SendInvoicePOLL (aiogram.enums.upe attribute), 463 tribute), 485	date_type.UpdateType at-
photo_height (aiogram.types.inline_query_result_photo.Inploid@aiorgRusultyPhotoxterna. attribute), 264 attribute), 130	
photo_height (aiogram.types.input_invoice_message_content_Inquitigram:eMessage&age attribute), 273 poll (aiogram.types.update.	Update attribute), 299
photo_size(aiogram.methods.create_invoice_link.CreateInolikaElnks in aiogram.types	
attribute), 460 POLL_ANSWER (aiogram.enu	ums.update_type.UpdateType
photo_size (aiogram.methods.send_invoice.SendInvoice attribute), 485	1 . X 1
attribute), 462 poll_answer (aiogram.type	
photo_size(aiogram.types.input_invoice_message_content.InputInva@MessageContent.attribute), 273 poll_id (aiogram.types.p	
photo_url (aiogram.methods.create_invoice_link.CreateInvoiceLink tribute), 212	ou_unswer.1 ouAnswer - ut-
attribute), 460 PollAnswer (class in aiogra	um.types.poll_answer), 212
photo_url (aiogram.methods.send_invoice.SendInvoice PollOption (class in aiogram.methods.send_invoice.SendInvoice PollOption)	
attribute), 462 PollType (class in aiogram.	
photo_url (aiogram.types.inline_query_result_photo.Inlin <b>e@siryiRrx.ultiDkoto</b> m.methods attribute), 263 attribute), 316	.set_sticker_position_in_set.SetStickerPositio
$photo\_url (\it aiogram.types.input\_invoice\_message\_content. \\ \textit{possistivo} i (\it elbersonza Grassenza Gras$	me_high_score.GameHighScore
attribute), 273 attribute), 302	
tribute), 576 tribute), 220	s.text_quote.TextQuote at-
photo_url (aiogram.utils.web_app.WebAppUser at- post_code (aiogram.types.st. tribute), 576 attribute), 295	
photo_width (aiogram.methods.create_invoice_link.CreateIRE(iaidginkm.enums.message attribute), 460 attribute), 481	
	ormatting), 586 gram.enums.update_type.UpdateType
photo_width(aiogram.types.inline_query_result_photo.InlineQueryResulbnPh)xt485 attribute), 263 pre_checkout_query(aiog	
photo_width(aiogram.types.input_invoice_message_content.InputInwohatMessageContent.inputInwohatMessage	
attribute), 222 attribute), 456	answer_pre_checkout_query.AnswerPreChec
PhotoSize (class in aiogram.types.photo_size), 211 PreCheckoutQuery	(class in
	checkout_query), 294
pin_message() (aiogram.types.chat.Chat method), 45 attribute), 152	ram.types.link_preview_options.LinkPreview
PinChatMessage (class in prefer_small_media(aiog aiogram.methods.pin_chat_message), 366 attribute), 151	gram.types.link_preview_options.LinkPreview
PINNED_MESSAGE (aiogram.enums.content_type.ContentTyperefix (aiogram.filters.con attribute), 474 tribute), 500	nmand.CommandObject at-
pinned_message (aiogram.types.chat.Chat attribute), PREMIUM (aiogram.enums.cha	at boost source type.ChatBoostSourceType

```
attribute), 472
                                                                                                                                                                      attribute), 385
premium_animation (aiogram.types.sticker.Sticker at- protect_content(aiogram.methods.send_document.SendDocument
                                                                                                                                                                      attribute), 387
                       tribute), 279
premium_subscription_month_count
                                                                                                                                              protect\_content (aiogram.methods.send\_game.SendGame
                        (aiogram.types.giveaway.Giveaway attribute),
                                                                                                                                                                       attribute), 453
                                                                                                                                              protect\_content (aiogram.methods.send\_invoice.SendInvoice
premium_subscription_month_count
                                                                                                                                                                      attribute), 463
                       (aiogram.types.giveaway_winners.GiveawayWinnprotect_content (aiogram.methods.send_location.SendLocation
                       attribute), 136
                                                                                                                                                                      attribute), 390
prepare_value() (aiogram.client.session.base.BaseSessiopprotect_content (aiogram.methods.send_media_group.SendMediaGroup
                       method), 15
                                                                                                                                                                      attribute), 392
prices (aiogram.methods.create_invoice_link.CreateInvoicpbintlect_content (aiogram.methods.send_message.SendMessage
                       attribute), 459
                                                                                                                                                                      attribute), 394
prices (aiogram.methods.send_invoice.SendInvoice at- protect_content(aiogram.methods.send_photo.SendPhoto
                                                                                                                                                                       attribute), 397
                        tribute), 462
prices (aiogram.types.input_invoice_message_content.InputProvaiceMessatgenChatingram.methods.send_poll.SendPoll
                       attribute), 273
                                                                                                                                                                      attribute), 400
prices (aiogram.types.shipping_option.ShippingOption protect_content(aiogram.methods.send_sticker.SendSticker
                       attribute), 296
                                                                                                                                                                      attribute), 311
PRIVATE (aiogram.enums.chat_type.ChatType attribute), protect_content (aiogram.methods.send_venue.SendVenue
                                                                                                                                                                      attribute), 402
prize_description(aiogram.types.giveaway.Giveaway protect_content(aiogram.methods.send_video.SendVideo
                        attribute), 135
                                                                                                                                                                       attribute), 405
prize_description (aiogram.types.giveaway_winners.Giparotæcklinamtent (aiogram.methods.send_video_note.SendVideoNote
                       attribute), 137
                                                                                                                                                                      attribute), 408
profile_accent_color_id (aiogram.types.chat.Chat protect_content(aiogram.methods.send_voice.SendVoice
                       attribute), 39
                                                                                                                                                                      attribute), 410
profile_accent_color_id
                                                                                                                                              \verb"provider_data" (a iogram.methods.create_invoice\_link.CreateInvoiceLink) \\
                       (aiogram.types.chat_full_info.ChatFullInfo
                                                                                                                                                                      attribute), 459
                       attribute), 58
                                                                                                                                              provider_data(aiogram.methods.send_invoice.SendInvoice
profile_background_custom_emoji_id
                                                                                                                                                                       attribute), 462
                        (aiogram.types.chat.Chat attribute), 39
                                                                                                                                              provider_data(aiogram.types.input_invoice_message_content.InputInvoi
profile_background_custom_emoji_id
                                                                                                                                                                      attribute), 273
                       (aiogram.types.chat_full_info.ChatFullInfo
                                                                                                                                              provider_payment_charge_id
                       attribute), 58
                                                                                                                                                                       (aiogram.types.successful_payment.SuccessfulPayment
promote() (aiogram.types.chat.Chat method), 46
                                                                                                                                                                      attribute), 297
PromoteChatMember
                                                                                                                                   in provider_token(aiogram.methods.create_invoice_link.CreateInvoiceLink
                        aiogram.methods.promote_chat_member),
                                                                                                                                                                      attribute), 459
                                                                                                                                              provider_token(aiogram.methods.send_invoice.SendInvoice
protect_content(aiogram.methods.copy_message.CopyMessage attribute), 462
                                                                                                                                              provider_token(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.InputInvoi
                       attribute), 330
protect_content (aiogram.methods.copy_messages.CopyMessages attribute), 273
                                                                                                                                              proximity_alert_radius
                       attribute), 332
{\tt protect\_content} \ (aiogram.methods.forward\_message.ForwardMes \center{Message} ageram.methods.edit\_message\_live\_location.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation.EditMessageLiveLocation
                                                                                                                                                                      attribute), 440
                       attribute), 346
protect_content (aiogram.methods.forward_messages.FapnourilMessagedert_radius
                                                                                                                                                                      (aiogram.methods.send_location.SendLocation
                       attribute), 347
protect_content (aiogram.methods.send_animation.SendAnimationuttribute), 389
                       attribute), 376
                                                                                                                                              proximity_alert_radius
                                                                                                                                                                       (aiogram.types.inline_query_result_location.InlineQueryResultLo
protect\_content (aiogram.methods.send\_audio.SendAudio
                                                                                                                                                                       attribute), 260
                       attribute), 379
protect_content (aiogram.methods.send_contact.SendCapataoximity_alert_radius
                       attribute), 383
                                                                                                                                                                      (aiogram.types.input_location_message_content.InputLocationMessage_content.InputLocationMessage_content.InputLocationMessage_content.InputLocationMessage_content.InputLocationMessage_content.InputLocationMessage_content.InputLocationMessage_content.InputLocationMessage_content.InputLocationMessage_content.InputLocationMessage_content.InputLocationMessage_content.InputLocationMessage_content.InputLocationMessage_content.InputLocationMessage_content.InputLocationMessage_content.InputLocationMessage_content.InputLocationMessage_content.InputLocationMessage_content.InputLocationMessage_content.InputLocationMessage_content.InputLocationMessage_content.InputLocationMessage_content.InputLocationMessage_content.InputLocationMessage_content.InputLocationMessage_content.InputLocationMessage_content.InputLocationMessage_content.InputLocationMessage_content.InputLocationMessage_content.InputLocationMessage_content.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.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.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
```

attribute), 275

protect\_content(aiogram.methods.send\_dice.SendDice

proximity_alert_radius	ReactionCount (class in
(aiogram.types.location.Location attribute), 152	aiogram.types.reaction_count), 214
PROXIMITY_ALERT_TRIGGERED	reactions (aiogram.types.message_reaction_count_updated.MessageRe attribute), 210
(aiogram.enums.content_type.ContentType attribute), 474	ReactionType (class in aiogram.types.reaction_type), 214
<pre>proximity_alert_triggered</pre>	ReactionTypeCustomEmoji (class in
(aiogram.types.message.Message attribute), 161	aiogram.types.reaction_type_custom_emoji), 214
ProximityAlertTriggered (class in	ReactionTypeEmoji (class in
aiogram.types.proximity_alert_triggered),	aiogram.types.reaction_type_emoji), 215
213	ReactionTypeType (class in
PYG (aiogram.enums.currency.Currency attribute), 477 Python Enhancement Proposals	aiogram.enums.reaction_type_type), 483 read() (aiogram.types.input_file.BufferedInputFile
PEP 484, 3	method), 139
PEP 492, 3	read() (aiogram.types.input_file.FSInputFile method),
111 132, 3	139
Q	read() (aiogram.types.input_file.InputFile method), 139
QAR (aiogram.enums.currency.Currency attribute), 477	read() (aiogram.types.input_file.URLInputFile method),
query (aiogram.types.chosen_inline_result.ChosenInlineR	esult 139
attribute), 229	receiver (aiogram.utils.web_app.WebAppInitData at-
query (aiogram.types.inline_query.InlineQuery at-	tribute), 574
tribute), 230	RECORD_VIDEO (aiogram.enums.chat_action.ChatAction
query (aiogram.types.switch_inline_query_chosen_chat.Sv attribute), 219	record_video() (aiogram.utils.chat_action.ChatActionSender
<pre>query_id (aiogram.utils.web_app.WebAppInitData at-</pre>	class method), 571
tribute), 574	RECORD_VIDEO_NOTE (aiogram.enums.chat_action.ChatAction
question (aiogram.methods.send_poll.SendPoll at-	attribute), 472
tribute), 399	record_video_note()
question (aiogram.types.poll.Poll attribute), 211	(aiogram.utils.chat_action.ChatActionSender
question_entities(aiogram.methods.send_poll.SendPo	RECORD_VOICE (aiogram.enums.chat_action.ChatAction
attribute), 399 question_entities (aiogram.types.poll.Poll attribute),	attribute), 472
212	${\tt record\_voice()} \ (a iogram.utils.chat\_action.ChatActionSender$
question_parse_mode	class method), 571  RED (aiogram.enums.topic_icon_color.TopicIconColor
(aiogram.methods.send_poll.SendPoll at- tribute), 399	attribute), 484
QUIZ (aiogram.enums.keyboard_button_poll_type_type.Ke	
attribute), 480	regexp_match(aiogram.filters.command.CommandObject
QUIZ (aiogram.enums.poll_type.PollType attribute), 483	attribute), 500
quote (aiogram.types.message.Message attribute), 158	register() (aiogram.fsm.scene.SceneRegistry method),
<pre>quote(aiogram.types.reply_parameters.ReplyParameters</pre>	546
attribute), 217	$\verb"register"() (a iogram. we bhook. a iohttp\_server. Base Request Handler$
<pre>quote_entities (aiogram.types.reply_parameters.ReplyI attribute), 217</pre>	Parameters method), 514 register() (aiogram.webhook.aiohttp_server.SimpleRequestHandler
${\tt quote\_parse\_mode} \ (a iogram. types. reply\_parameters. Rep$	lyParamete <b>rs</b> ethod), 514
attribute), 217	register() (aiogram.webhook.aiohttp_server.TokenBasedRequestHandl
${\tt quote\_position} \ (a iogram. types. reply\_parameters. Reply Label{thm:parameters}) \\$	Parameters method), 515
attribute), 217	REGULAR (aiogram.enums.keyboard_button_poll_type_type.KeyboardButt
D	attribute), 480
R	REGULAR (aiogram.enums.poll_type.PollType attribute), 483
react() (aiogram.types.message.Message method), 204	
reaction (aiogram.methods.set_message_reaction.SetMesattribute), 420	sageReaction (atogram.enums.sucker_type.suckerType attribute), 483

- remove\_caption(aiogram.methods.copy\_messages.Copy\_messages attribute), 332 attribute), 383
- remove\_date (aiogram.types.chat boost removed.ChatBoosephryomackup (aiogram.methods.send dice.SendDice attribute), 53 attribute), 385
- remove\_keyboard (aiogram.types.reply\_keyboard\_remove\_keplyKmdxkudRaingvam.methods.send\_document.SendDocument attribute), 387 attribute), 216

attribute), 463

attribute), 394

attribute), 397

tribute), 400

attribute), 311

attribute), 402

attribute), 405

attribute), 408

attribute), 410

attribute), 446

tribute), 447

attribute), 233

attribute), 235

attribute), 237

attribute), 239

attribute), 241

attribute), 243

attribute), 245

attribute), 247

 $reply_markup\ (aiogram.methods.send_location.SendLocation$ 

 $\verb"reply_markup" (a iogram.methods.send_photo.SendPhoto")$ 

 $\verb"reply_markup" (a iogram.methods.send_sticker.SendSticker")$ 

reply\_markup(aiogram.methods.send\_venue.SendVenue

reply\_markup(aiogram.methods.send\_video.SendVideo

reply\_markup (aiogram.methods.send\_voice.SendVoice

reply\_markup (aiogram.methods.stop\_poll.StopPoll at-

reply\_markup(aiogram.methods.send\_video\_note.SendVideoNote

reply\_markup(aiogram.methods.stop\_message\_live\_location.StopMessag

reply\_markup(aiogram.types.inline\_query\_result\_article.InlineQueryRes

reply\_markup(aiogram.types.inline\_query\_result\_audio.InlineQueryResu

reply\_markup(aiogram.types.inline\_query\_result\_cached\_audio.InlineQu

- REMOVED\_CHAT\_BOOST (aiogram.enums.update type.Updata\*Expley\_markup (aiogram.methods.send game.SendGame attribute), 485 attribute), 453
- removed\_chat\_boost (aiogram.types.update.Update at- reply\_markup (aiogram.methods.send\_invoice.SendInvoice tribute), 300
- render() (aiogram.utils.formatting.Text method), 583
- RENTAL\_AGREEMENT (aiogram.enums.encrypted\_passport\_element.Encatrythatel)assportElement attribute), 479 reply\_markup(aiogram.methods.send\_message.SendMessage
- ReopenForumTopic (class in aiogram.methods.reopen\_forum\_topic), 370
- ReopenGeneralForumTopic (class inaiogram.methods.reopen\_general\_forum\_topic), reply\_markup (aiogram.methods.send\_poll.SendPoll at-371
- ReplaceStickerInSet (class aiogram.methods.replace\_sticker\_in\_set),
- reply() (aiogram.types.message.Message method), 180
- reply\_animation() (aiogram.types.message.Message *method*), 163
- reply\_audio() (aiogram.types.message.Message method), 166
- reply\_contact() (aiogram.types.message.Message *method*), 168
- reply\_dice() (aiogram.types.message.Message method), 186
- reply\_document() (aiogram.types.message.Message method), 170
- reply\_game() (aiogram.types.message.Message *method*), 172
- reply\_invoice() (aiogram.types.message.Message *method*), 173
- reply\_location() (aiogram.types.message.Message
- method), 176
- reply\_markup(aiogram.methods.copy\_message.CopyMessngply\_markup(aiogram.types.inline\_query\_result\_cached\_document.Inlin attribute), 330
- reply\_markup(aiogram.methods.edit\_message\_caption.Edit#\$\disparackappi\langlemiogram.types.inline\_query\_result\_cached\_gif.InlineQuer attribute), 438
- reply\_markup(aiogram.methods.edit\_message\_live\_locationpfdjt\_Markuyo\_Laivet\_roomtispes.inline\_query\_result\_cached\_mpeg4\_gif.Inli attribute), 440
- reply\_markup (aiogram.methods.edit\_message\_media.EditMetpslsqqmMekdiqp (aiogram.types.inline\_query\_result\_cached\_photo.InlineQi attribute), 441
- reply\_markup (aiogram.methods.edit\_message\_reply\_marknepp ExliptNacksrape(RinglyNMourkpes.inline\_query\_result\_cached\_sticker.InlineQ
- attribute), 443 reply\_markup (aiogram.methods.edit\_message\_text.EditMr.sspltyTmarkup (aiogram.types.inline\_query\_result\_cached\_video.InlineQu
- attribute), 376

attribute), 444

- reply\_markup(aiogram.methods.send\_audio.SendAudio reply\_markup(aiogram.types.inline\_query\_result\_contact.InlineQueryRe attribute), 379
- attribute), 249 reply\_markup (aiogram.methods.send\_animation.SendAnimemplyn\_markup (aiogram.types.inline\_query\_result\_cached\_voice.InlineQu attribute), 251

attribute), 253

```
reply_markup(aiogram.types.inline_query_result_document.InlineQueribles)ulfDocument
         attribute), 255
                                                      reply_parameters(aiogram.methods.send video note.SendVideoNote
reply_markup(aiogram.types.inline_query_result_game.InlineQueryResiblt@yr408
                                                      \verb"reply_parameters" (a iogram. methods. send\_voice. Send Voice
         attribute), 256
reply_markup(aiogram.types.inline_query_result_gif.InlineQueryResmttGife), 410
        attribute), 258
                                                      reply_photo()
                                                                             (aiogram.types.message.Message
reply_markup(aiogram.types.inline query result location.InlineOuwnetRodultIntantial
                                                      reply_poll()
         attribute), 260
                                                                             (aiogram.types.message.Message
reply_markup(aiogram.types.inline_query_result_mpeg4_gif.InlineQuethoResultMpeg4Gif
                                                      reply_sticker()
         attribute), 262
                                                                             (aiogram.types.message.Message
reply_markup(aiogram.types.inline_query_result_photo.InlineQuery_ResholdP.hbb8
         attribute), 264
                                                      reply_to_message
                                                                            (aiogram.types.message.Message
reply_markup (aiogram.types.inline_query_result_venue.InlineQuery_ResultYep\text{ie})
                                                      reply_to_message_id
        attribute), 266
reply_markup(aiogram.types.inline_query_result_video.InlineQueryRixxydtWidnothods.copy_message.CopyMessage
         attribute), 268
                                                                attribute), 330
reply_markup(aiogram.types.inline_query_result_voice.IndiaprDuery_Rnands klopiceid
         attribute), 270
                                                                (aiogram.methods.send animation.SendAnimation
reply_markup (aiogram.types.message.Message
                                                                attribute), 376
         tribute), 162
                                                      reply_to_message_id
reply_media_group()
                                                                (aiogram.methods.send_audio.SendAudio
         (aiogram.types.message.Message
                                            method),
                                                                attribute), 379
         178
                                                      reply_to_message_id
reply_parameters(aiogram.methods.copy message.CopyMessage (aiogram.methods.send contact.SendContact
                                                                attribute), 383
         attribute), 330
reply_parameters(aiogram.methods.send_animation.Sendepilynation_message_id
         attribute), 376
                                                                (aiogram.methods.send_dice.SendDice
                                                                                                        at-
\verb"reply_parameters" (a io gram. methods. send\_audio. Send Audio
                                                                tribute), 385
        attribute), 379
                                                      reply_to_message_id
                                                                (aiogram.methods.send_document.SendDocument
reply_parameters (aiogram.methods.send_contact.SendContact
         attribute), 383
                                                                attribute), 387
reply_parameters(aiogram.methods.send_dice.SendDicereply_to_message_id
                                                                (aiogram.methods.send_game.SendGame
         attribute), 385
reply_parameters (aiogram.methods.send_document.SendDocumentttribute), 453
         attribute), 387
                                                      reply_to_message_id
reply_parameters(aiogram.methods.send_game.SendGame
                                                                (aiogram.methods.send invoice.SendInvoice
        attribute), 453
                                                                attribute), 463
reply_parameters(aiogram.methods.send_invoice.SendInvepdey_to_message_id
         attribute), 463
                                                                (aiogram.methods.send location.SendLocation
reply_parameters (aiogram.methods.send_location.SendLocation attribute), 390
                                                      reply_to_message_id
        attribute), 390
reply_parameters(aiogram.methods.send_media_group.SendMediaGroupm.methods.send_media_group.SendMediaGroup
                                                                attribute), 392
         attribute), 392
reply_parameters(aiogram.methods.send_message.Send\textbygeto_message_id
                                                                (aiogram.methods.send_message.SendMessage
         attribute), 394
                                                                attribute), 394
reply_parameters (aiogram.methods.send_photo.SendPhoto
         attribute), 397
                                                      reply_to_message_id
reply_parameters(aiogram.methods.send_poll.SendPoll
                                                                (aiogram.methods.send_photo.SendPhoto
         attribute), 400
                                                                attribute), 397
reply_parameters(aiogram.methods.send_sticker.SendStickeply_to_message_id
                                                                (aiogram.methods.send_poll.SendPoll
        attribute), 311
                                                                                                        at-
                                                                tribute), 400
reply_parameters(aiogram.methods.send_venue.SendVenue
                                                      reply_to_message_id
         attribute), 402
                                                                (aiogram.methods.send sticker.SendSticker
reply_parameters (aiogram.methods.send video.SendVideo
```

attribute), 311	request_photo (aiogram.types.keyboard_button_request_users.Keyboard
reply_to_message_id (aiogram.methods.send_venue.SendVenue	attribute), 151 request_pol1 (aiogram.types.keyboard_button.KeyboardButton
attribute), 402	attribute), 147
reply_to_message_id	$\verb"request_title" (a iogram. types. keyboard\_button\_request\_chat. Keyboard\_button\_request\_chat.$
$(a iogram.methods.send\_video.Send Video$	attribute), 149
attribute), 405	request_user(aiogram.types.keyboard_button.KeyboardButton
reply_to_message_id	attribute), 147
(atogram.metnoas.sena_viaeo_note.SenaviaeoNo attribute), 408	<pre>meequest_username(aiogram.types.keyboard_button_request_chat.Keyboo attribute), 149</pre>
reply_to_message_id	$\verb"request_username" (a iogram. types. keyboard\_button\_request\_users. Keyboard\_button\_reques$
$(a iogram.methods.send\_voice.Send Voice$	attribute), 151
attribute), 410	request_users (aiogram.types.keyboard_button.KeyboardButton
reply_to_story (aiogram.types.message.Message at-	attribute), 147
tribute), 159	request_write_access
reply_venue() (aiogram.types.message.Message method), 189	(aiogram.types.login_url.LoginUrl attribute), 153
reply_video() (aiogram.types.message.Message	
method), 191	(aiogram.fsm.scene.SceneConfig attribute),
reply_video_note() (aiogram.types.message.Message	547
	reset_history_on_enter
reply_voice() (aiogram.types.message.Message method), 195	(aiogram.fsm.scene.SceneConfig attribute), 547
	$\verb"resize_keyboard" (a iogram. types. reply_keyboard\_markup. Reply Keyboard\_markup. Reply $
aiogram.utils.keyboard), 564	attribute), 215
	resolve_bot() (aiogram.webhook.aiohttp_server.BaseRequestHandler
aiogram.types.reply_keyboard_markup),	method), 514
215	resolve_bot() (aiogram.webhook.aiohttp_server.SimpleRequestHandler
ReplyKeyboardRemove (class in aiogram.types.reply_keyboard_remove), 216	method), 514 resolve_bot() (aiogram.webhook.aiohttp_server.TokenBasedRequestHar
ReplyParameters (class in	method), 515
	resolve_used_update_types()
request_chat(aiogram.types.keyboard_button.Keyboard.	
attribute), 147	490
request_contact (aiogram.types.keyboard_button.Keybo	
attribute), 147 request_id (aiogram.types.chat_shared.ChatShared at-	aiogram.types.response_parameters), 218  RestartingTelegram 554
tribute), 126	restrict() (aiogram.types.chat.Chat method), 47
request_id(aiogram.types.keyboard_button_request_cha	
attribute), 148	aiogram.methods.restrict_chat_member),
request_id(aiogram.types.keyboard_button_request_user	
	RESTRICTED (aiogram.enums.chat_member_status.ChatMemberStatus
<pre>request_id(aiogram.types.keyboard_button_request_user</pre>	rs.Keyboard <b>Brittat</b> -BegันิestUsers
attribute), 150	$\verb"result" (a iogram.methods.answer\_web\_app\_query. AnswerWebAppQuery$
request_id (aiogram.types.user_shared.UserShared at-	attribute), 451
tribute), 223	result_id(aiogram.types.chosen_inline_result.ChosenInlineResult
request_id (aiogram.types.users_shared.UsersShared	attribute), 229
attribute), 223	results (aiogram.methods.answer_inline_query.AnswerInlineQuery
request_location (aiogram.types.keyboard_button.Keyb	
attribute), 147	retake() (aiogram.fsm.scene.SceneWizard method), 548 sæt.KeybbuedRuitogrkogstels.response_parameters.ResponseParameters
attribute), 151	attribute), 218
	MEN HRSBostib ButtiogRaque suChsupassport_element_error_type.PassportEle
attribute), 149	attribute), 482

reverse_side (aiogram.types.encrypted_passport_eleme attribute), 283	n <b>sEherspiwe</b> Passpo( <b>tliberrem</b> ttypes.force_reply.ForceReply attribute), 132
	selective (aiogram.types.reply_keyboard_markup.ReplyKeyboardMarkup attribute), 216
	Ba <b>sa Dear Meva</b> lkaiogram.types.reply_keyboard_remove.ReplyKeyboardRemove attribute), 216
	SELFIE (aiogram.enums.passport_element_error_type.PassportElementErrattribute), 482
373	$\verb selfie  (a iogram.types.encrypted\_passport\_element.EncryptedPassportElement.EncryptedPasspor$
rights (aiogram.methods.set_my_default_administrator_r attribute), 422	rights.SetM <b>ythefauleAdin</b> inistratorRights send_copy() (aiogram.types.message.Message method),
RON (aiogram.enums.currency.Currency attribute), 477	197
ROSE (aiogram.enums.topic_icon_color.TopicIconColor attribute), 484	send_email_to_provider (aiogram.methods.create_invoice_link.CreateInvoiceLink
rotation_angle (aiogram.types.background_fill_gradien	
attribute), 21	send_email_to_provider
Router (class in aiogram.dispatcher.router), 489	(aiogram.methods.send_invoice.SendInvoice
row() (aiogram.utils.keyboard.InlineKeyboardBuilder	attribute), 463
method), 564	send_email_to_provider
row() (aiogram.utils.keyboard.ReplyKeyboardBuilder method), 565	(aiogram.types.input_invoice_message_content.InputInvoiceMessattribute), 274
RSD (aiogram.enums.currency.Currency attribute), 477	send_phone_number_to_provider
RUB (aiogram.enums.currency.Currency attribute), 477	(aiogram.methods.create_invoice_link.CreateInvoiceLink
run_polling() (aiogram.dispatcher.dispatcher.Dispatche	er attribute), 460
method), 496	send_phone_number_to_provider
0	(aiogram.methods.send_invoice.SendInvoice
S	attribute), 463
<pre>safe_parse_webapp_init_data() (in module</pre>	send_phone_number_to_provider
aiogram.utils.web_app), 573	(aiogram.types.input_invoice_message_content.InputInvoiceMess
SAR (aiogram.enums.currency.Currency attribute), 477	attribute), 274
scale (aiogram.types.mask_position.MaskPosition attribute), 278	SendAnimation (class in aiogram.methods.send_animation), 375
Scene (class in aiogram.fsm.scene), 545	SendAudio (class in aiogram.methods.send_audio), 377
SceneConfig (class in aiogram.fsm.scene), 546	SendChatAction (class in
SceneException, 554	aiogram.methods.send_chat_action), 380
SceneRegistry (class in aiogram.fsm.scene), 545	SendContact (class in aiogram.methods.send_contact),
ScenesManager (class in aiogram.fsm.scene), 546	382
SceneWizard (class in aiogram.fsm.scene), 547	SendDice (class in aiogram.methods.send_dice), 384
${\tt scope} \ (aiogram.methods.delete\_my\_commands.DeleteMy Commands.DeleteMy Command$	CSandDocument (class in
attribute), 339	aiogram.methods.send_document), 386
attribute), 357	ases DER (aiogram.enums.chat_type.ChatType attribute), 473
scope (aiogram.methods.set_my_commands.SetMyComma attribute), 421	attribute), 158
score (aiogram.methods.set_game_score.SetGameScore attribute), 455	sender_business_bot (aiogram.types.message.Message attribute),
score (aiogram.types.game_high_score.GameHighScore attribute), 302	158 sender_chat (aiogram.types.message.Message at-
secret (aiogram.types.encrypted_credentials.EncryptedC attribute), 281	
secret_token(aiogram.methods.set_webhook.SetWebhook	17
attribute), 468 SEK (aiogram.enums.currency.Currency attribute), 477	sender_chat_id(aiogram.methods.ban_chat_sender_chat.BanChatSende attribute), 325
San (mogramichums.current y.Current y unitoute), 4//	sender_chat_id(aiogram.methods.unban_chat_sender_chat.UnbanChat

attribute), 428	SetChatPermissions (class	in
sender_user (aiogram.types.message_origin_user.Messagattribute), 209	415	
sender_user_name (aiogram.types.message_origin_hidde attribute), 208	er <b>Sats@h&amp;fPskage</b> OriginHiddenUs <b>(</b> xlass aiogram.methods.set_chat_photo), 416	in
SendGame (class in aiogram.methods.send_game), 452	SetChatStickerSet (class	in
SendInvoice (class in aiogram.methods.send_invoice), 461	<pre>aiogram.methods.set_chat_sticker_set), 417 SetChatTitle (class</pre>	in
SendLocation (class in	aiogram.methods.set_chat_title), 418	
$aiogram.methods.send\_location),388$	SetCustomEmojiStickerSetThumbnail (class	in
SendMediaGroup (class in	aiogram.methods.set_custom_emoji_sticker	set_thumbnail),
aiogram.methods.send_media_group), 391	312	
SendMessage (class in aiogram.methods.send_message),	`	in
393	aiogram.methods.set_game_score), 454	
SendPhoto (class in aiogram.methods.send_photo), 395	SetMessageReaction (class	in
SendPol1 (class in aiogram.methods.send_poll), 398	aiogram.methods.set_message_reaction),	
SendSticker (class in aiogram.methods.send_sticker),	419	
310	SetMyCommands (class	in
SendVenue (class in aiogram.methods.send_venue), 401	<pre>aiogram.methods.set_my_commands), 421 SetMyDefaultAdministratorRights (class</pre>	:
SendVideo (class in aiogram.methods.send_video), 403 SendVideoNote (class in	SetMyDefaultAdministratorRights (class aiogram.methods.set_my_default_administra	in tor violeta)
SendVideoNote (class in aiogram.methods.send_video_note), 406	422	ior_rignis),
SendVoice (class in aiogram.methods.send_voice), 409	SetMyDescription (class	in
SentWebAppMessage (class in closes), 409	aiogram.methods.set_my_description), 423	ın
aiogram.types.sent_web_app_message), 277	SetMyName (class in aiogram.methods.set_my_nam	e)
set_administrator_custom_title()	424	c),
(aiogram.types.chat.Chat method), 45	SetMyShortDescription (class	in
set_data() (aiogram.fsm.scene.SceneWizard method),	aiogram.methods.set_my_short_description),	
548	425	
set_data() (aiogram.fsm.storage.base.BaseStorage	SetPassportDataErrors (class	in
method), 531	aiogram.methods.set_passport_data_errors),	
<pre>set_description() (aiogram.types.chat.Chat method),</pre>	469	
49	SetStickerEmojiList (class	in
<pre>set_locale() (aiogram.utils.i18n.middleware.FSMI18nM method), 568</pre>	Aiddleware aiogram.methods.set_sticker_emoji_list), 313	
set_name (aiogram.types.sticker.Sticker attribute), 279	SetStickerKeywords (class	in
<pre>set_permissions() (aiogram.types.chat.Chat method), 46</pre>	<pre>aiogram.methods.set_sticker_keywords), 314</pre>	
<pre>set_photo() (aiogram.types.chat.Chat method), 49</pre>	SetStickerMaskPosition (class	in
<pre>set_position_in_set() (aiogram.types.sticker.Sticker method), 280</pre>	<pre>aiogram.methods.set_sticker_mask_position) 315</pre>	,
set_state() (aiogram.fsm.storage.base.BaseStorage	SetStickerPositionInSet (class	in
method), 530	aiogram.methods.set_sticker_position_in_set	),
<pre>set_sticker_set() (aiogram.types.chat.Chat method),</pre>	316	
43	SetStickerSetThumbnail (class	in
set_title() (aiogram.types.chat.Chat method), 49	aiogram.methods.set_sticker_set_thumbnail),	,
SetChatAdministratorCustomTitle (class in	317	
aiogram.methods.set_chat_administrator_custom		in
411	aiogram.methods.set_sticker_set_title), 319	
SetChatDescription (class in	setup() (aiogram.utils.i18n.middleware.I18nMiddlew	are
aiogram.methods.set_chat_description), 413	method), 569	1 \
SetChatMenuButton (class in	SetWebhook (class in aiogram.methods.set_webhook	к),
aiogram.methods.set_chat_menu_button),	467	
414	SGD (aiogram.enums.currency.Currency attribute), 477	

```
SharedUser (class in aiogram.types.shared_user), 218
                                                                                                                                                                                                                                attribute), 53
shifted_id (aiogram.types.chat.Chat property), 39
                                                                                                                                                                                                source(aiogram.types.chat_boost_source_gift_code.ChatBoostSourceGift
shipping_address(aiogram.types.order_info.OrderInfo
                                                                                                                                                                                                                                attribute), 54
                               attribute), 293
                                                                                                                                                                                                source (aiogram.types.chat_boost_source_giveaway.ChatBoostSourceGive
shipping_address (aiogram.types.shipping_query.ShippingQuery attribute), 55
                               attribute), 296
                                                                                                                                                                                                source (aiogram.types.chat_boost_source_premium.ChatBoostSourcePrem
shipping_option_id(aiogram.types.pre_checkout_query.PreCheckotttQuate), 55
                                attribute), 294
                                                                                                                                                                                                 source (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
shipping_option_id(aiogram.types.successful_payment.SuccessfulPtryinetet), 284
                               attribute), 297
                                                                                                                                                                                                source (aiogram.types.passport_element_error_file.PassportElementError
shipping_options (aiogram.methods.answer_shipping_query.AnswatShipping_Query
                               attribute), 457
                                                                                                                                                                                                source (aiogram.types.passport_element_error_files.PassportElementErro
SHIPPING_QUERY (aiogram.enums.update_type.UpdateType
                                                                                                                                                                                                                                attribute), 286
                               attribute), 485
                                                                                                                                                                                                source (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.PassportEl
shipping_query (aiogram.types.update.Update
                                                                                                                                                                                                                                attribute), 287
                                tribute), 299
                                                                                                                                                                                                source (aiogram.types.passport_element_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_reverse_side.PassportElement_error_revers
shipping_query_id (aiogram.methods.answer_shipping_query.Answershipp)ing@uery
                               attribute), 457
                                                                                                                                                                                                source (aiogram.types.passport_element_error_selfie.PassportElementErr
ShippingAddress
                                                                                                                                                                                 in
                                                                                                               (class
                                                                                                                                                                                                                                 attribute), 289
                               aiogram.types.shipping_address), 295
                                                                                                                                                                                                source (aiogram.types.passport_element_error_translation_file.PassportE
ShippingOption
                                                                                                             (class
                                                                                                                                                                                 in
                                                                                                                                                                                                                                attribute), 289
                               aiogram.types.shipping_option), 295
                                                                                                                                                                                                source (aiogram.types.passport_element_error_translation_files.Passport)
ShippingQuery
                                                                                                                                                                                                                                 attribute), 290
                                                                                                           (class
                                                                                                                                                                                 in
                               aiogram.types.shipping_query), 296
                                                                                                                                                                                                source (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_e
short_description(aiogram.methods.set_my_short_description.SetMs;iShoor)Description
                                                                                                                                                                                               {\tt SPOILER} \ (a iogram.enums.message\_entity\_type.MessageEntityType
                               attribute), 425
short_description(aiogram.types.bot_short_description.BotShortDtesibnipe)on481
                                                                                                                                                                                               Spoiler (class in aiogram.utils.formatting), 585
                               attribute), 30
show_above_text (aiogram.types.link_preview_options.LisklanvielaOp(ioiogram.types.video_chat_scheduled.VideoChatScheduled
                               attribute), 152
                                                                                                                                                                                                                                attribute), 226
show_alert (aiogram.methods.answer_callback_query.AnssteacCallback@u&nyogram.utils.web_app.WebAppInitData
                               attribute), 321
                                                                                                                                                                                                                                attribute), 575
show_alert(aiogram.utils.callback_answer.CallbackAnswertart_parameter(aiogram.methods.send_invoice.SendInvoice
                               property), 579
                                                                                                                                                                                                                                attribute), 462
SimpleI18nMiddleware
                                                                                                                         (class
                                                                                                                                                                                               start_parameter(aiogram.types.inline_query_results_button.InlineQuer
                               aiogram.utils.i18n.middleware), 567
                                                                                                                                                                                                                                attribute), 270
SimpleRequestHandler
                                                                                                                                                                                               start_parameter
                                                                                                                                                                                                                                                                                     (aiogram.types.invoice.Invoice
                                aiogram.webhook.aiohttp_server), 514
                                                                                                                                                                                                                                 attribute), 292
SLOT_MACHINE (aiogram.enums.dice_emoji.DiceEmoji
                                                                                                                                                                                               start_polling() (aiogram.dispatcher.dispatcher.Dispatcher
                               attribute), 478
                                                                                                                                                                                                                                method), 496
SLOT_MACHINE
                                                              (aiogram.types.dice.DiceEmoji
                                                                                                                                                                                               state (aiogram.fsm.scene.SceneConfig attribute), 547
                                tribute), 128
                                                                                                                                                                                                state(aiogram.types.shipping_address.ShippingAddress
slow_mode_delay (aiogram.types.chat.Chat attribute),
                                                                                                                                                                                                                               attribute), 295
                                                                                                                                                                                               STATIC
                                                                                                                                                                                                                                    (aiogram.enums.sticker_format.StickerFormat
\verb|slow_mode_delay| (a iogram.types.chat_full_info.ChatFullInfo|
                                                                                                                                                                                                                                attribute), 483
                                attribute), 59
                                                                                                                                                                                                status (aiogram.types.chat_member_administrator.ChatMemberAdministr
\verb|small_file_id| (a iogram.types.chat\_photo.ChatPhoto|
                                                                                                                                                                                                                                attribute), 100
                               attribute), 125
                                                                                                                                                                                                status (aiogram.types.chat_member_banned.ChatMemberBanned
small_file_unique_id
                                                                                                                                                                                                                                attribute), 102
                                (aiogram.types.chat_photo.ChatPhoto
                                                                                                                                                                                               status(aiogram.types.chat_member_left.ChatMemberLeft
                               tribute), 125
                                                                                                                                                                                                                                attribute), 102
source (aiogram.types.chat_boost.ChatBoost attribute),
                                                                                                                                                                                               status(aiogram.types.chat\_member\_member.ChatMemberMember
                                                                                                                                                                                                                                attribute), 103
{\tt source}\ (aiogram.types.chat\_boost\_removed.ChatBoostRem{\textbf{stat}} \\ {\tt us}\ (aiogram.types.chat\_member\_owner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwner.ChatMemberOwn
```

```
tribute), 280
              attribute), 103
status (aiogram.types.chat_member_restricted.ChatMemb&rtlaskoir&et (class in aiogram.types.sticker_set), 280
                                                                                     StickerType (class in aiogram.enums.sticker type), 483
             attribute), 104
STICKER (aiogram.enums.content_type.ContentType at-
                                                                                    stop_live_location()
              tribute), 473
                                                                                                   (aiogram.types.message.Message
                                                                                                                                                          method),
STICKER (aiogram.enums.inline query result type.InlineQueryResultI) pe
              attribute), 479
                                                                                     stop_polling() (aiogram.dispatcher.dispatcher.Dispatcher
sticker(aiogram.methods.add\_sticker\_to\_set.AddStickerToSet
                                                                                                   method), 496
              attribute), 303
                                                                                     StopMessageLiveLocation
                                                                                                                                             (class
                                                                                                                                                                    in
sticker (aiogram.methods.delete_sticker_from_set.DeleteStickerFromSegram.methods.stop_message_live_location),
              attribute), 305
sticker (aiogram.methods.replace_sticker_in_set.Replace\$\text{StrdpPbi\}\text{El}(class in aiogram.methods.stop_poll), 447
                                                                                     STORY (aiogram.enums.content_type.ContentType at-
             attribute), 309
sticker (aiogram.methods.send_sticker.SendSticker at-
                                                                                                   tribute), 473
              tribute), 310
                                                                                     {\tt story}\ (aiogram.types.external\_reply\_info.ExternalReplyInfo
sticker(aiogram.methods.set_sticker_emoji_list.SetStickerEmojiListattribute), 130
              attribute), 313
                                                                                     story (aiogram.types.message.Message attribute), 159
sticker (aiogram.methods.set_sticker_keywords.SetSticker Ktowyrdslass in aiogram.types.story), 219
              attribute), 314
                                                                                     stream_content() (aiogram.client.session.base.BaseSession
sticker (aiogram.methods.set sticker mask position.SetStickerMaskReditid), 15
              attribute), 315
                                                                                     street_line1(aiogram.types.shipping_address.ShippingAddress
sticker (aiogram.methods.set_sticker_position_in_set.SetStickerPosititoribuSet), 295
                                                                                     street_line2(aiogram.types.shipping_address.ShippingAddress
              attribute), 316
sticker(aiogram.methods.upload sticker file.UploadStickerFile
                                                                                                   attribute), 295
                                                                                     STRIKETHROUGH (aiogram.enums.message_entity_type.MessageEntityType
              attribute), 320
sticker (aiogram.types.business_intro.BusinessIntro at-
                                                                                                   attribute), 481
              tribute), 31
                                                                                     Strikethrough (class in aiogram.utils.formatting), 585
\verb|sticker|| (a iogram. types. external\_reply\_info. ExternalReply \textit{BM} \texttt{CESSFUL\_PAYMENT}|| (a iogram. enums. content\_type. ContentType) | (a iogram. enums. content\_type. ContentType) | (a iogram. enums. external\_reply\_info. ExternalReply \textit{BM} \texttt{CESSFUL\_PAYMENT}|| (a iogram. enums. enu
             attribute), 129
                                                                                                   attribute), 474
                       (aiogram.types.input_sticker.InputSticker
                                                                                     successful_payment (aiogram.types.message.Message
sticker
              attribute), 277
                                                                                                   attribute), 161
sticker (aiogram.types.message.Message attribute), 159 SuccessfulPayment
                                                                                                                                        (class
                                                                                                                                                                    in
Sticker (class in aiogram.types.sticker), 279
                                                                                                   aiogram.types.successful_payment), 297
sticker_file_id(aiogram.types.inline_query_result_cackedges.tedIntine_Query_ResultCachedSticker
                                                                                                   (aiogram.methods.create invoice link.CreateInvoiceLink
              attribute), 247
sticker_format(aiogram.methods.create new sticker set.CreateNewtSibuken)Set59
             attribute), 304
                                                                                     suggested_tip_amounts
sticker_format (aiogram.methods.upload_sticker_file.UploadSticketdfidgram.methods.send_invoice.SendInvoice
              attribute), 320
                                                                                                   attribute), 462
sticker_set_name(aiogram.methods.set_chat_sticker_setsSugGbatStickerisetamounts
             attribute), 417
                                                                                                   (aiogram.types.input invoice message content.InputInvoiceMess
sticker_set_name (aiogram.types.chat.Chat attribute),
                                                                                                   attribute), 273
                                                                                     SUPERGROUP
                                                                                                                   (aiogram.enums.chat type.ChatType
sticker_set_name(aiogram.types.chat_full_info.ChatFullInfo
                                                                                                   attribute), 473
                                                                                     SUPERGROUP_CHAT_CREATED
              attribute), 60
sticker_type (aiogram.methods.create_new_sticker_set.CreateNew.Stiicker_type.ContentType
                                                                                                   attribute), 474
              attribute), 304
sticker_type (aiogram.types.sticker_set.StickerSet at- supergroup_chat_created
              tribute), 280
                                                                                                   (aiogram.types.message.Message
                                                                                                                                                        attribute),
StickerFormat
                                               (class
                                                                              in
             aiogram.enums.sticker_format), 483
                                                                                     supports_inline_queries (aiogram.types.user.User
stickers (aiogram.methods.create_new_sticker_set.CreateNewStickersetbute), 221
              attribute), 304
                                                                                     supports\_streaming(aiogram.methods.send\_video.SendVideo)
                   (aiogram.types.sticker set.StickerSet
                                                                                                   attribute), 405
stickers
                                                                             at-
```

```
supports_streaming(aiogram.types.input_media_video.lupntMaidigNidetypes.inline_query_results_button.InlineQueryResultsButton
                   attribute), 146
                                                                                                                                         attribute), 270
switch_inline_query
                                                                                                                      text (aiogram.types.input poll option.InputPollOption
                    (aiogram.types.inline_keyboard_button.InlineKeyboardButtontribute), 146
                   attribute), 138
                                                                                                                      text (aiogram.types.keyboard_button.KeyboardButton
switch_inline_query_chosen_chat
                                                                                                                                          attribute), 146
                   (aiogram.types.inline keyboard button.InlineKeybaxxdButtaniogram.types.menu button.MenuButton
                   attribute), 138
                                                                                                                                         tribute), 154
switch_inline_query_current_chat
                                                                                                                      text(aiogram.types.menu_button_web_app.MenuButtonWebApp
                   (aiogram.types.inline_keyboard_button.InlineKeyboardButtantribute), 155
                   attribute), 138
                                                                                                                     text (aiogram.types.message.Message attribute), 159
switch_pm_parameter
                                                                                                                      text (aiogram.types.poll_option.PollOption attribute),
                   (aiogram.methods.answer_inline_query.AnswerInlineQuery213
                   attribute), 449
                                                                                                                      text (aiogram.types.text_quote.TextQuote attribute), 220
\verb|switch_pm_text| (a iogram.methods.answer_inline\_query. \verb|AtexxterInl| (a iogram.methods.answer\_inline\_query. AttexxterInl| (a 
                   attribute), 449
                                                                                                                                         property), 579
SwitchInlineQueryChosenChat
                                                                                                             in Text (class in aiogram.utils.formatting), 583
                                                                                   (class
                   aiogram.types.switch_inline_query_chosen_chat),text_entities (aiogram.types.game.Game attribute),
                   219
                                                                                                                                         302
                                                                                                                      text_entities(aiogram.types.input poll option.InputPollOption
                                                                                                                                         attribute), 146
                                                                                                                      text_entities (aiogram.types.poll_option.PollOption
telegram_payment_charge_id
                                                                                                                                         attribute), 213
                    (aiogram.types.successful payment.SuccessfulPayment
                                                                                                                      TEXT_LINK (aiogram.enums.message_entity_type.MessageEntityType
                   attribute), 297
                                                                                                                                         attribute), 481
TelegramAPIError, 554
TelegramAPIServer (class in aiogram.client.telegram),
                                                                                                                     TEXT_MENTION (aiogram.enums.message_entity_type.MessageEntityType
                                                                                                                                          attribute), 481
                                                                                                                      text_parse_mode(aiogram.types.input_poll_option.InputPollOption
TelegramBadRequest, 554
                                                                                                                                         attribute), 146
TelegramConflictError, 554
                                                                                                                     TextLink (class in aiogram.utils.formatting), 586
TelegramEntityTooLarge, 554
TelegramForbiddenError, 554
                                                                                                                     TextMention (class in aiogram.utils.formatting), 586
TelegramMigrateToChat, 554
                                                                                                                     TextQuote (class in aiogram.types.text_quote), 220
                                                                                                                     THB (aiogram.enums.currency.Currency attribute), 477
TelegramNetworkError, 554
                                                                                                                      theme_name(aiogram.types.background_type_chat_theme.BackgroundType
TelegramNotFound, 554
TelegramRetryAfter, 554
                                                                                                                                          attribute), 22
                                                                                                                      thumbnail (aiogram.methods.send animation.SendAnimation
TelegramServerError, 554
TelegramUnauthorizedError, 554
                                                                                                                                         attribute), 376
                                                                                                                      thumbnail (aiogram.methods.send_audio.SendAudio at-
TEMPORARY_REGISTRATION
                    (aiogram.enums.encrypted_passport_element.EncryptedPastpbutE)enent
                   attribute), 479
                                                                                                                      thumbnail ({\it aiogram.methods.send\_document.SendDocument}
                                                                                                                                         attribute), 387
TEXT
                 (aiogram.enums.content_type.ContentType
                                                                                                                      thumbnail\ (aiogram.methods.send\_video.SendVideo\ at-
                    tribute), 473
                                                                                                                                         tribute), 404
text (aiogram.filters.command.CommandObject prop-
                                                                                                                      \verb|thumbnail| (a iogram.methods.send\_video\_note.SendVideoNote|
                   erty), 500
{\tt text} (a iogram. methods. answer\_callback\_query. Answer Callback\_Quer {\tt optimization}, 407 answer {\tt optimization}, 407 answer
                                                                                                                      thumbnail(aiogram.methods.set_sticker_set_thumbnail.SetStickerSetThum
                    attribute), 321
\texttt{text}(aiogram.methods.edit\_message\_text.EditMessageText)
                                                                                                                                         attribute), 318
                                                                                                                      thumbnail
                                                                                                                                                   (aiogram.types.animation.Animation
                   attribute), 444
{\tt text} (aiogram.methods.send_message.SendMessage at-
                                                                                                                                         tribute), 19
                    tribute), 393
                                                                                                                      thumbnail (aiogram.types.audio.Audio attribute), 20
                                                                                                                      thumbnail
                                                                                                                                                   (aiogram.types.document.Document
text (aiogram.types.game.Game attribute), 301
text(aiogram.types.inline_keyboard_button.InlineKeyboardButton tribute), 128
                                                                                                                      thumbnail ({\it aiogram.types.input\_media\_animation.InputMediaAnimation}) \\
                   attribute), 138
                                                                                                                                          attribute), 140
```

attribute), 260

```
thumbnail (aiogram.types.input_media_audio.InputMediaAhhhhhobnail_width (aiogram.types.inline_query_result_venue.InlineQuery_
                                         attribute), 142
                                                                                                                                                                                                                                                                                                       attribute), 266
thumbnail (aiogram.types.input_media_document.InputMedim@ozomentname (aiogram.types.business_opening_hours.BusinessOpenin
                                          attribute), 143
                                                                                                                                                                                                                                                                                                       attribute), 32
thumbnail (aiogram.types.input_media_video.InputMedia\titmeout (aiogram.methods.get_updates.GetUpdates at-
                                         attribute), 145
                                                                                                                                                                                                                                                                                                       tribute), 466
thumbnail (aiogram.types.sticker.Sticker attribute), 279
                                                                                                                                                                                                                                                            {\tt title}\ (aiogram.methods.create\_invoice\_link.CreateInvoiceLink
thumbnail (aiogram.types.sticker_set.StickerSet at-
                                                                                                                                                                                                                                                                                                       attribute), 459
                                         tribute), 281
                                                                                                                                                                                                                                                            \verb+title+ (a iogram.methods.create\_new\_sticker\_set.CreateNewStickerSet
thumbnail (aiogram.types.video.Video attribute), 225
                                                                                                                                                                                                                                                                                                       attribute), 304
thumbnail (aiogram.types.video_note.VideoNote at- title
                                                                                                                                                                                                                                                                                                     (aiogram.methods.send_audio.SendAudio
                                          tribute), 227
                                                                                                                                                                                                                                                                                                       tribute), 379
thumbnail_height(aiogram.types.inline_query_result_artiitelknli(vei@gwaryNeveulhtvAlsiseled_invoice.SendInvoice at-
                                         attribute), 233
                                                                                                                                                                                                                                                                                                       tribute), 461
thumbnail_height (aiogram.types.inline_query_result_comiacted nline@geannementlebanteentd_venue.SendVenue at-
                                         attribute), 253
                                                                                                                                                                                                                                                                                                       tribute), 402
thumbnail_height (aiogram.types.inline_query_result_docintelent.lalingomerrellesodktDocanantitle.SetChatTitle at-
                                         attribute), 255
                                                                                                                                                                                                                                                                                                      tribute), 418
thumbnail_height (aiogram.types.inline_query_result_locatedne_hullionerQuery\textracker_set_title.SetStickerSetTitle
                                          attribute), 260
                                                                                                                                                                                                                                                                                                       attribute), 319
thumbnail_height (aiogram.types.inline_query_result_vertice. Ireliant Qureum Republi Mendine. Audio attribute), 19
                                         attribute), 266
                                                                                                                                                                                                                                                            title (aiogram.types.business_intro.BusinessIntro at-
                                                                                                                                                                                                                                                                                                       tribute), 31
thumbnail_mime_type
                                          (aiogram.types.inline_query_result_gif.InlineQueryResalt@iifgram.types.chat.Chat attribute), 35
                                         attribute), 258
                                                                                                                                                                                                                                                            title (aiogram.types.chat_full_info.ChatFullInfo at-
thumbnail_mime_type
                                                                                                                                                                                                                                                                                                      tribute), 57
                                          (aiogram.types.inline_query_result_mpeg4_gif.InltinaQaeryRaialyMpeg4f6ifchat_shared.ChatShared
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  at-
                                         attribute), 262
                                                                                                                                                                                                                                                                                                       tribute), 126
thumbnail_url (aiogram.types.inline_query_result_articlet.htdr:QuegyRuxtrlp?sstjalme.Game attribute), 301
                                                                                                                                                                                                                                                            attribute), 233
thumbnail_url(aiogram.types.inline_query_result_contact.InlineQuatriRuse)tContact
                                         attribute), 253
                                                                                                                                                                                                                                                            \verb|title| (a iogram.types.inline\_query\_result\_audio.InlineQueryResultAudio|)|
thumbnail_url(aiogram.types.inline_query_result_document.Inline@twibytRe)xullDocument
                                          attribute), 255
                                                                                                                                                                                                                                                            {\tt title}\ (aiogram.types.inline\_query\_result\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQueryResult\_cached\_document.InlineQuer
thumbnail_url(aiogram.types.inline_query_result_gif.InlineQueryRatsihGtif), 239
                                         attribute), 257
                                                                                                                                                                                                                                                            title(aiogram.types.inline_query_result_cached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQueryResultCached_gif.InlineQ
thumbnail_url (aiogram.types.inline_query_result_location.InlineQuateri/Ruesi).lfl&cation
                                          attribute), 260
                                                                                                                                                                                                                                                            title(aiogram.types.inline_query_result_cached_mpeg4_gif.InlineQuery)
thumbnail_url(aiogram.types.inline_query_result_mpeg4_gif.InlineQuibuxResultMpeg4Gif
                                                                                                                                                                                                                                                            {\tt title}\ (aiogram.types.inline\_query\_result\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.InlineQueryResult\_cached\_photo.I
                                          attribute), 262
thumbnail_url(aiogram.types.inline_query_result_photo.InlineQueantRibsudleP,ho45
                                          attribute), 263
                                                                                                                                                                                                                                                            {\tt title}\ (aiogram.types.inline\_query\_result\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.InlineQueryResult\_cached\_video.I
thumbnail_url(aiogram.types.inline_query_result_venue.InlineQuearttRibsult) & Ale
                                         attribute), 266
                                                                                                                                                                                                                                                            \verb|title| (a iogram.types.in line\_query\_result\_cached\_voice.In lineQueryResult\_cached\_voice.In lineQueryResult\_cached\_voice.I
thumbnail_url(aiogram.types.inline_query_result_video.InlineQueryfkilsuleVid&d
                                         attribute), 268
                                                                                                                                                                                                                                                            title(aiogram.types.inline_query_result_document.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocument.InlineQueryResultDocu
thumbnail_width(aiogram.types.inline_query_result_article.InlineQtteibyRe)yuTArticle
                                         attribute), 233
                                                                                                                                                                                                                                                            title(aiogram.types.inline_query_result_gif.InlineQueryResultGif
thumbnail_width (aiogram.types.inline_query_result_contact.Inline@urebyRe)sulfContact
                                         attribute), 253
                                                                                                                                                                                                                                                            \verb|title| (a iogram.types.inline\_query\_result\_location.InlineQueryResultLocation.InlineQueryResultLocation.InlineQueryResultLocation.InlineQueryResultLocation.InlineQueryResultLocation.InlineQueryResultLocation.InlineQueryResultLocation.InlineQueryResultLocation.InlineQueryResultLocation.InlineQueryResultLocation.InlineQueryResultLocation.InlineQueryResultLocation.InlineQueryResultLocation.InlineQueryResultLocation.InlineQueryResultLocation.InlineQueryResultLocation.InlineQueryResultLocation.InlineQueryResultLocation.InlineQueryResultLocation.InlineQueryResultLocation.InlineQueryResultLocation.InlineQueryResultLocation.InlineQueryResultLocation.InlineQueryResultLocation.InlineQueryResultLocation.InlineQueryResultLocation.InlineQueryResultLocation.InlineQueryResultLocation.InlineQueryResultLocation.InlineQueryResultLocation.InlineQueryResultLocation.InlineQueryResultLocation.InlineQueryResultLocation.InlineQueryResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResultResul
thumbnail_width(aiogram.types.inline_query_result_document.InlinuteQbuere)ResultDocument
                                         attribute), 255
                                                                                                                                                                                                                                                            title(aiogram.types.inline_query_result_mpeg4_gif.InlineQueryResultMy
thumbnail_width(aiogram.types.inline_query_result_location.Inlina@nibryRes266Location
```

704 Index

title(aiogram.types.inline query result photo.InlineQueryResultPhoto

```
attribute), 264
                                                                                                                                                                                                     attribute), 21
title (aiogram.types.inline_query_result_venue.InlineQuertyffes.utilventum.types.background_fill_gradient.BackgroundFillGradient
                           attribute), 265
                                                                                                                                                                                                     attribute), 21
title (aiogram.types.inline_query_result_video.InlineQuery.Resultticleam.types.background_fill_solid.BackgroundFillSolid
                            attribute), 268
                                                                                                                                                                                                      attribute), 21
title (aiogram.types.inline_query_result_voice.InlineQuery; RepulaVajcam.types.background_type_chat_theme.BackgroundTypeChatTh
                           attribute), 269
                                                                                                                                                                                                      attribute), 22
title (aiogram.types.input_invoice_message_content.Input!hypri(eMogramgargnestentckground_type_fill.BackgroundTypeFill
                            attribute), 272
                                                                                                                                                                                                     attribute), 23
\verb|title| (aiogram.types.input\_media\_audio.InputMediaAudiotype| (aiogram.types.background\_type\_pattern.BackgroundTypePattern| (aiogram.types.background\_types.background\_type\_pattern.Background\_type\_pattern.Background\_type\_pattern.Background\_type\_pattern.Background\_type\_pattern.Background\_type\_pattern.Background\_type\_pattern.Background\_type\_pattern.Background\_type\_pattern.Background\_type\_pattern.Background\_type\_pattern.Background\_type\_pattern.Background\_type\_pattern.Background\_type\_pattern.Background\_type\_pattern.Background\_type\_pattern.Background\_type\_pattern.Background\_type\_pattern.Background\_type\_pattern.Background\_type\_pattern.Background\_type\_pattern.Background\_type\_pattern.Background\_type\_pattern.Background\_type\_pattern.Background\_type\_pattern.Background\_type\_pattern.Background\_type\_pattern.Background\_type\_pattern.Background\_type\_pattern.Background\_type\_pattern.Background\_type\_pattern.Background\_type\_pattern.Background\_type\_pattern.Background\_type\_pattern.Background\_type\_pattern.Background\_type\_pattern.Background\_type\_pattern.Backgro
                            attribute), 142
                                                                                                                                                                                                     attribute), 23
title (aiogram.types.input_venue_message_content.InputVzypeMaixgyae6.ovpentbackground_type_wallpaper.BackgroundTypeWallpaper
                           attribute), 276
                                                                                                                                                                                                     attribute), 24
title (aiogram.types.invoice.Invoice attribute), 292
                                                                                                                                                                         type (aiogram.types.bot_command_scope_all_chat_administrators.BotCom
title (aiogram.types.shipping_option.ShippingOption
                                                                                                                                                                                                     attribute), 26
                            attribute), 295
                                                                                                                                                                         type (aiogram.types.bot_command_scope_all_group_chats.BotCommandS
title (aiogram.types.sticker_set.StickerSet attribute),
                                                                                                                                                                                                     attribute), 26
                                                                                                                                                                         type (aiogram.types.bot_command_scope_all_private_chats.BotCommand
title (aiogram.types.venue.Venue attribute), 224
                                                                                                                                                                                                     attribute), 27
title (aiogram.utils.web_app.WebAppChat attribute),
                                                                                                                                                                         type (aiogram.types.bot_command_scope_chat.BotCommandScopeChat
                                                                                                                                                                                                     attribute), 27
TJS (aiogram.enums.currency.Currency attribute), 477
                                                                                                                                                                         type (aiogram.types.bot_command_scope_chat_administrators.BotComma
TokenBasedRequestHandler
                                                                                                                                                                                                     attribute), 28
                                                                                                                  (class
                                                                                                                                                            in
                            aiogram.webhook.aiohttp_server), 515
                                                                                                                                                                         type (aiogram.types.bot_command_scope_chat_member.BotCommandScop
top_color (aiogram.types.background_fill_gradient.BackgroundFill@attrdbeute), 29
                           attribute), 21
                                                                                                                                                                         type (aiogram.types.bot_command_scope_default.BotCommandScopeDefa
TopicIconColor
                                                                                                                                                            in
                                                                                                                                                                                                      attribute), 29
                                                                                               (class
                                                                                                                                                                         type (aiogram.types.chat.Chat attribute), 35
                           aiogram.enums.topic_icon_color), 484
total_amount (aiogram.types.invoice.Invoice attribute),
                                                                                                                                                                         type (aiogram.types.chat_background.ChatBackground
                                                                                                                                                                                                      attribute), 52
total_amount(aiogram.types.pre_checkout_query.PreChetkppetQueoriogram.types.chat_full_info.ChatFullInfo
                           attribute), 294
                                                                                                                                                                                                     tribute), 57
total_amount (aiogram.types.successful_payment.Successful_payment.types.encrypted_passport_element.EncryptedPassportElement.
                            attribute), 297
                                                                                                                                                                                                     attribute), 282
total_count (aiogram.types.reaction_count.ReactionCounttype (aiogram.types.inline_query_result_article.InlineQueryResultArticle
                                                                                                                                                                                                     attribute), 233
                           attribute), 214
total_count (aiogram.types.user_profile_photos.UserProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfilephotos.userProfileph
                            attribute), 222
                                                                                                                                                                                                     attribute), 234
total_voter_count (aiogram.types.poll.Poll attribute), type (aiogram.types.inline_query_result_cached_audio.InlineQueryResult
                                                                                                                                                                                                     attribute), 236
translation (a iogram. types. encrypted\_passport\_element. \textit{Expression} that a partial properties \textit{Proposition} that
                            attribute), 283
                                                                                                                                                                                                      attribute), 239
TRANSLATION_FILE (aiogram.enums.passport_element_erroypts/paibgssport/besinehtler.queliyperesult_cached_gif.InlineQueryResultCac
                           attribute), 482
                                                                                                                                                                                                     attribute), 240
TRANSLATION_FILES (aiogram.enums.passport_element_eruppet/wpia/PassptynpEsianleneEquerTyppesult_cached_mpeg4_gif.InlineQueryRi
                            attribute), 482
                                                                                                                                                                                                      attribute), 243
traveler (aiogram.types.proximity_alert_triggered.Proximitypalentaliregaeradpes.inline_query_result_cached_photo.InlineQueryResult
                            attribute), 213
                                                                                                                                                                                                     attribute), 245
TRY (aiogram.enums.currency.Currency attribute), 477
                                                                                                                                                                         type (aiogram.types.inline_query_result_cached_sticker.InlineQueryResult
TTD (aiogram.enums.currency.Currency attribute), 477
                                                                                                                                                                                                      attribute), 247
TWD (aiogram.enums.currency.Currency attribute), 477
                                                                                                                                                                         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), 248
                                                                                                                                                                         type (aiogram.types.inline_query_result_cached_voice.InlineQueryResultC
```

type (aiogram.types.background fill freeform gradient.BackgroundFallthFileufe)m260adient

```
type (aiogram.types.inline query result contact.InlineOuetyRes(diGentanctypes.passport element error files.PassportElementErrorF
                                 attribute), 252
                                                                                                                                                                                                                                         attribute), 286
type (aiogram.types.inline query result document.Inline Otorpe Rasiole Proactive perspective) element error front side. Passport Element E
                                 attribute), 254
                                                                                                                                                                                                                                         attribute), 287
 type (aiogram.types.inline_query_result_game.InlineQueryRypal(Gioggaam.types.passport_element_error_reverse_side.PassportElement
                                 attribute), 256
                                                                                                                                                                                                                                         attribute), 288
type (aiogram.types.inline query result gif.InlineOueryRexylp6(faiogram.types.passport element error selfie.PassportElementError.
                                  attribute), 257
                                                                                                                                                                                                                                         attribute), 289
type (aiogram.types.inline_query_result_location.InlineQuery,Pesalthywantionpes.passport_element_error_translation_file.PassportElement_error_translation_file.PassportElement_error_translation_file.PassportElement_error_translation_file.PassportElement_error_translation_file.PassportElement_error_translation_file.PassportElement_error_translation_file.PassportElement_error_translation_file.PassportElement_error_translation_file.PassportElement_error_translation_file.PassportElement_error_translation_file.PassportElement_error_translation_file.PassportElement_error_translation_file.PassportElement_error_translation_file.PassportElement_error_translation_file.PassportElement_error_translation_file.PassportElement_error_translation_file.PassportElement_error_translation_file.PassportElement_error_translation_file.PassportElement_error_translation_file.PassportElement_error_translation_file.PassportElement_error_translation_file.PassportElement_error_translation_file.PassportElement_error_translation_file.PassportElement_error_translation_file.PassportElement_error_translation_file.PassportElement_error_translation_file.PassportElement_error_translation_file.PassportElement_error_translation_file.PassportElement_error_translation_file.PassportElement_error_translation_file.PassportElement_error_translation_file.PassportElement_error_translation_file.PassportElement_error_translation_file.PassportElement_error_translation_file.PassportElement_error_translation_file.PassportElement_error_translation_file.PassportElement_error_translation_file.PassportElement_error_translation_file.PassportElement_error_translation_file.PassportElement_error_translation_file.PassportElement_error_translation_file.PassportElement_error_translation_file.PassportElement_error_translation_file.PassportElement_error_translation_file.PassportElement_error_translation_file.PassportElement_error_translation_file.PassportElement_error_translation_file.PassportElement_error_translation_file.PassportElement_error_translation_file.PassportElement
                                 attribute), 259
                                                                                                                                                                                                                                         attribute), 289
type (aiogram.types.inline_query_result_mpeg4_gif.Inline Qvypy ReindtMpag#Eipassport_element_error_translation_files.PassportEle
                                 attribute), 261
                                                                                                                                                                                                                                         attribute), 290
type (aiogram.types.inline_query_result_photo.InlineQueryRepallatibanam.types.passport_element_error_unspecified.PassportElement
                                 attribute), 263
                                                                                                                                                                                                                                         attribute), 291
{\tt type}\ (aiogram.types.inline\_query\_result\_venue.InlineQuery {\tt type}\ ({\tt diagram.types.poll.Poll}\ attribute),\ 212
                                  attribute), 265
                                                                                                                                                                                                        type (aiogram.types.reaction_count.ReactionCount at-
type (aiogram.types.inline_query_result_video.InlineQueryResultVideabute), 214
                                                                                                                                                                                                        type (aiogram.types.reaction_type_custom_emoji.ReactionTypeCustomEmo
                                 attribute), 267
type (aiogram.types.inline_query_result_voice.InlineQueryResultVoiatribute), 214
                                                                                                                                                                                                        type (aiogram.types.reaction_type_emoji.ReactionTypeEmoji
                                  attribute), 269
type (aiogram.types.input_media_animation.InputMediaAnimation attribute), 215
                                 attribute), 140
                                                                                                                                                                                                        type (aiogram.types.sticker.Sticker attribute), 279
 type(aiogram.types.input_media_audio.InputMediaAudio type (aiogram.utils.web_app.WebAppChat attribute),
                                  attribute), 141
type (aiogram.types.input media document.InputMediaDoTNPhENG
                                                                                                                                                                                                                                           (aiogram.enums.chat action.ChatAction
                                 attribute), 143
                                                                                                                                                                                                                                         tribute), 471
 type(aiogram.types.input_media_photo.InputMediaPhoto_typing() (aiogram.utils.chat_action.ChatActionSender
                                                                                                                                                                                                                                         class method), 571
                                  attribute), 144
 type (aiogram.types.input_media_video.InputMediaVideo TZS (aiogram.enums.currency.Currency attribute), 477
                                 attribute), 145
 type (aiogram.types.keyboard_button_poll_type.KeyboardButtonPollType
                                 attribute), 147
                                                                                                                                                                                                       UAH (aiogram.enums.currency.Currency attribute), 477
                               (aiogram.types.menu_button.MenuButton
 type
                                                                                                                                                                                                       UGX (aiogram.enums.currency.Currency attribute), 477
                                 tribute), 154
                                                                                                                                                                                                       unban() (aiogram.types.chat.Chat method), 48
type(aiogram.types.menu_button_commands.MenuButtonGammands_MenuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonGammands_menuButtonG
                                                                                                                                                                                                                                                                                                                  (aiogram.types.chat.Chat
                                 attribute), 154
                                                                                                                                                                                                                                         method), 40
(class
                                                                                                                                                                                                                                                                                                                                                                                                in
                                  attribute), 155
                                                                                                                                                                                                                                         aiogram.methods.unban_chat_member),
type (aiogram.types.menu_button_web_app.MenuButtonWebApp
                                                                                                                                                                                                                                         426
                                 attribute), 155
                                                                                                                                                                                                        UnbanChatSenderChat
                                                                                                                                                                                                                                                                                                                                   (class
                                                                                                                                                                                                                                                                                                                                                                                               in
type (aiogram.types.message entity.MessageEntity at-
                                                                                                                                                                                                                                         aiogram.methods.unban_chat_sender_chat),
                                  tribute), 205
                                                                                                                                                                                                                                         428
type (aiogram.types.message_origin_channel.MessageOriginChanneld_prize_count
                                 attribute), 207
                                                                                                                                                                                                                                         (aiogram.types.giveaway completed.GiveawayCompleted
{\tt type} \ (aiogram. types. message\_origin\_chat. MessageOriginChat
                                                                                                                                                                                                                                         attribute), 135
                                  attribute), 208
                                                                                                                                                                                                        unclaimed_prize_count
{\tt type} \ (aiogram.types.message\_origin\_hidden\_user. MessageOriginHidden\_itser. MessageOriginHidden
                                 attribute), 208
                                                                                                                                                                                                                                         attribute), 137
{\tt type}\ (aiogram. types. message\_origin\_user. MessageOriginU_{\tt DMDERLINE}\ (aiogram. enums. message\_entity\_type. MessageEntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeAntityTypeA
                                  attribute), 209
                                                                                                                                                                                                                                         attribute), 481
type (aiogram.types.passport_element_error_data_field.PassportFlementEasorPatoField.utils.formatting), 585
                                 attribute), 284
                                                                                                                                                                                                        UnhideGeneralForumTopic
{\tt type} \ (aiogram. types. passport\_element\_error\_file. PassportElementError_file. PassportElementE
                                 attribute), 285
                                                                                                                                                                                                                                         429
```

UPLOAD_PHOTO (aiogram.enums.chat_action.ChatAction attribute), 471
upload_photo() (aiogram.utils.chat_action.ChatActionSender class method), 571
UPLOAD_VIDEO (aiogram.enums.chat_action.ChatAction
attribute), 471
<pre>upload_video() (aiogram.utils.chat_action.ChatActionSender</pre>
class method), 571
UPLOAD_VIDEO_NOTE (aiogram.enums.chat_action.ChatAction
attribute), 472
<pre>upload_video_note()</pre>
(aiogram.utils.chat_action.ChatActionSender
class method), 571
UPLOAD_VOICE (aiogram.enums.chat_action.ChatAction
attribute), 472
gamp,load_voice() (aiogram.utils.chat_action.ChatActionSender
class method), 571
UploadStickerFile (class in
c_messagesgjogram.methods.upload_sticker_file), 320
URL (aiogram.enums.message_entity_type.MessageEntityType
attribute), 481
url (aiogram.methods.answer_callback_query.AnswerCallbackQuery
attribute), 321 url (aiogram.methods.set_webhook.SetWebhook at-
url (aiogram.methods.set_webhook.SetWebhook at- tribute), 468
url (aiogram.types.inline_keyboard_button.InlineKeyboardButton
attribute), 138
p <b>axP</b> ak <b>spogtdihettypetEirtlonTypa</b> ery_result_article.InlineQueryResultArticle
paxPaksipogtdibe.trypesEirtlon&yquery_result_article.InlineQueryResultArticle attribute), 233
pasPassipogradibe.tryepesEirolom&yquery_result_article.InlineQueryResultArticle attribute), 233 url (aiogram.types.link_preview_options.LinkPreviewOptions
parPakspogtdilettypesEirthonTypaery_result_article.InlineQueryResultArticle attribute), 233 url (aiogram.types.link_preview_options.LinkPreviewOptions utMember attribute), 151
parPa(sspogrative.trypesErrstonTypeery_result_article.InlineQueryResultArticle attribute), 233 url (aiogram.types.link_preview_options.LinkPreviewOptions attMember attribute), 151 url (aiogram.types.login_url.LoginUrl attribute), 153
parPaksipogidile.trypresEirnlömEygaery_result_article.InlineQueryResultArticle attribute), 233 url (aiogram.types.link_preview_options.LinkPreviewOptions utMember attribute), 151 url (aiogram.types.login_url.LoginUrl attribute), 153 riarChatMen(beogram.types.message_entity.MessageEntity attribute), 206
parPa(sajografilettypesEirthömEypetery_result_article.InlineQueryResultArticle attribute), 233 url (aiogram.types.link_preview_options.LinkPreviewOptions utMember attribute), 151 url (aiogram.types.login_url.LoginUrl attribute), 153 ritrChatMem(biogram.types.message_entity.MessageEntity attribute), 206 enable (Biogradn.types.user.User property), 221
parPa(sajografilettypesEirthömTypesery_result_article.InlineQueryResultArticle attribute), 233 url (aiogram.types.link_preview_options.LinkPreviewOptions utMember attribute), 151 url (aiogram.types.login_url.LoginUrl attribute), 153 riurChatMen(beogram.types.message_entity.MessageEntity
parPa(sajograficite.trypresEirchömTypaery_result_article.InlineQueryResultArticle attribute), 233 url (aiogram.types.link_preview_options.LinkPreviewOptions attMember attribute), 151 url (aiogram.types.login_url.LoginUrl attribute), 153 rixtChatMen(beogram.types.message_entity.MessageEntity attribute), 206 euibler(Biogram.types.user.User property), 221 url (aiogram.types.web_app_info.WebAppInfo attribute), MemberRestRicted
parfa(sipograficientypresEirchömTypeaery_result_article.InlineQueryResultArticle attribute), 233 url (aiogram.types.link_preview_options.LinkPreviewOptions attMember attribute), 151 url (aiogram.types.login_url.LoginUrl attribute), 153 rixtChatMen(hieogram.types.message_entity.MessageEntity attribute), 206 entitle(Biogram.types.user.User property), 221 url (aiogram.types.web_app_info.WebAppInfo attribute), MemberRestricted url (aiogram.types.webhook_info.WebhookInfo at-
parPa(sajograficientypresEirchömTypaery_result_article.InlineQueryResultArticle attribute), 233 url (aiogram.types.link_preview_options.LinkPreviewOptions attMember attribute), 151 url (aiogram.types.login_url.LoginUrl attribute), 153 riarChatMen(hieogram.types.message_entity.MessageEntity attribute), 206 enable(Biogram.types.user.User property), 221 url (aiogram.types.web_app_info.WebAppInfo attribute), MemberRestReted url (aiogram.types.webhook_info.WebhookInfo attribute), 300
parPa(sajografilettypesEirthömeypeaery_result_article.InlineQueryResultArticle attribute), 233 url (aiogram.types.link_preview_options.LinkPreviewOptions attMember attribute), 151 url (aiogram.types.login_url.LoginUrl attribute), 153 rirrChatMem(beogram.types.message_entity.MessageEntity attribute), 206 enable(Biogram.types.user.User property), 221 url (aiogram.types.web_app_info.WebAppInfo attribute), MemberRestReted url (aiogram.types.webhook_info.WebhookInfo attribute), 300 url (aiogram.utils.callback_answer.CallbackAnswer
url (aiogram.types.link_preview_options.LinkPreviewOptions uttribute), 233 url (aiogram.types.link_preview_options.LinkPreviewOptions utMember attribute), 151 url (aiogram.types.login_url.LoginUrl attribute), 153 ritrChatMen(biogram.types.message_entity.MessageEntity
url (aiogram.types.link_preview_options.LinkPreviewOptions uttribute), 233 url (aiogram.types.link_preview_options.LinkPreviewOptions utMember attribute), 151 url (aiogram.types.login_url.LoginUrl attribute), 153 riurChatMen(beogram.types.message_entity.MessageEntity
parPa(sajogntilentyppesEirnlömTypaery_result_article.InlineQueryResultArticle attribute), 233 url (aiogram.types.link_preview_options.LinkPreviewOptions attMember attribute), 151 url (aiogram.types.login_url.LoginUrl attribute), 153 riarChatMen(biogram.types.message_entity.MessageEntity attribute), 206 enable(Biognadn.types.user.User property), 221 url (aiogram.types.web_app_info.WebAppInfo attribute), MemberRestitcted url (aiogram.types.webhook_info.WebhookInfo attribute), 300 url (aiogram.utils.callback_answer.CallbackAnswer property), 580 Url (class in aiogram.utils.formatting), 584 URLInputFile (class in aiogram.types.input_file), 139,
parPa(sipogrativerypresErritionTypeaery_result_article.InlineQueryResultArticle attribute), 233 url (aiogram.types.link_preview_options.LinkPreviewOptions attMember attribute), 151 url (aiogram.types.login_url.LoginUrl attribute), 153 riarChatMen(biogram.types.message_entity.MessageEntity attribute), 206 enable(Biogradn.types.user.User property), 221 url (aiogram.types.web_app_info.WebAppInfo attribute), MemberRestricted url (aiogram.types.webhook_info.WebhookInfo attribute), 300 url (aiogram.utils.callback_answer.CallbackAnswer property), 580 Url (class in aiogram.utils.formatting), 584 URLInputFile (class in aiogram.types.input_file), 139, 488
parPa(sipogratile.typesEirnkinnTypeaery_result_article.InlineQueryResultArticle attribute), 233 url (aiogram.types.link_preview_options.LinkPreviewOptions attMember attribute), 151 url (aiogram.types.login_url.LoginUrl attribute), 153 riarChatMen(hieogram.types.message_entity.MessageEntity attribute), 206 enable(Biogram.types.user.User property), 221 url (aiogram.types.web_app_info.WebAppInfo attribute), MemberRestred url (aiogram.types.webhook_info.WebhookInfo attribute), 300 url (aiogram.utils.callback_answer.CallbackAnswer property), 580 Url (class in aiogram.utils.formatting), 584 URLInputFile (class in aiogram.types.input_file), 139, 488 USD (aiogram.enums.currency.Currency attribute), 477
parPa(sipogratile.typesEirnkinaTypeaery_result_article.InlineQueryResultArticle attribute), 233 url (aiogram.types.link_preview_options.LinkPreviewOptions attMember attribute), 151 url (aiogram.types.login_url.LoginUrl attribute), 153 riarChatMen(hieogram.types.message_entity.MessageEntity attribute), 206 enable(Biogram.types.user.User property), 221 url (aiogram.types.web_app_info.WebAppInfo attribute), MemberRestReted url (aiogram.types.webhook_info.WebhookInfo attribute), 300 url (aiogram.utils.callback_answer.CallbackAnswer property), 580 Url (class in aiogram.utils.formatting), 584 URLInputFile (class in aiogram.types.input_file), 139, 488 USD (aiogram.enums.currency.Currency attribute), 477 use_independent_chat_permissions
url (aiogram.types.link_preview_options.LinkPreviewOptions utMember attribute), 151 url (aiogram.types.login_url.LoginUrl attribute), 153 riurChatMen(biogram.types.message_entity.MessageEntity
parfacesportile tryenes Firston Eypaery_result_article.InlineQueryResultArticle attribute), 233 url (aiogram.types.link_preview_options.LinkPreviewOptions attMember attribute), 151 url (aiogram.types.login_url.LoginUrl attribute), 153 rittChatMen(beogram.types.message_entity.MessageEntity attribute), 206 enable (Biogram.types.user.User property), 221 url (aiogram.types.web_app_info.WebAppInfo attribute), MemberRestReted url (aiogram.types.webhook_info.WebhookInfo attribute), 300 url (aiogram.utils.callback_answer.CallbackAnswer property), 580 Url (class in aiogram.utils.formatting), 584 URLInputFile (class in aiogram.types.input_file), 139, 488 USD (aiogram.enums.currency.Currency attribute), 477 use_independent_chat_permissions
parfaceprotein types finion type arry_result_article.InlineQueryResultArticle attribute), 233  url (aiogram.types.link_preview_options.LinkPreviewOptions attMember attribute), 151  url (aiogram.types.login_url.LoginUrl attribute), 153  rittChatMen(hivogram.types.message_entity.MessageEntity attribute), 206  enable (Biogram.types.user.User property), 221  url (aiogram.types.web_app_info.WebAppInfo attribute), MemberRestPicted  url (aiogram.types.webhook_info.WebhookInfo attribute), 300  url (aiogram.utils.callback_answer.CallbackAnswer property), 580  Url (class in aiogram.utils.formatting), 584  URLInputFile (class in aiogram.types.input_file), 139, 488  USD (aiogram.enums.currency.Currency attribute), 477  use_independent_chat_permissions  (aiogram.methods.restrict_chat_member.RestrictChatMember attribute), 372  use_independent_chat_permissions
parPaksjingthetypersEintimFypaery_result_article.InlineQueryResultArticle attribute), 233 url (aiogram.types.link_preview_options.LinkPreviewOptions utMember attribute), 151 url (aiogram.types.login_url.LoginUrl attribute), 153 riutChatMen(biogram.types.message_entity.MessageEntity attribute), 206 enable(Biogram.types.user.User property), 221 url (aiogram.types.web_app_info.WebAppInfo attribute), MemberRestricted url (aiogram.types.webhook_info.WebhookInfo at- tribute), 300 url (aiogram.utils.callback_answer.CallbackAnswer property), 580 Url (class in aiogram.utils.formatting), 584 URLInputFile (class in aiogram.types.input_file), 139, 488 USD (aiogram.enums.currency.Currency attribute), 477 use_independent_chat_permissions (aiogram.methods.restrict_chat_member.RestrictChatMember attribute), 372 use_independent_chat_permissions (aiogram.methods.set_chat_permissions.SetChatPermissions)
parPassing the typers Finding ypaery_result_article. Inline Query Result Article attribute), 233  url (aiogram.types.link_preview_options. Link Preview Options at Member attribute), 151  url (aiogram.types.login_url. Login Url attribute), 153  ritt Chat Men (heorgram.types.message_entity. Message Entity attribute), 206  eutble (Biogram.types.user. User property), 221  url (aiogram.types.web_app_info. Web App Info attribute), Member Restricted  url (aiogram.types.webhook_info. Webhook Info attribute), 300  url (aiogram.utils.callback_answer. Callback Answer property), 580  Url (class in aiogram.utils.formatting), 584  URL Input File (class in aiogram.types.input_file), 139, 488  USD (aiogram.enums.currency. Currency attribute), 477  use_independent_chat_permissions
partaspingtabletypersEinton Typaery_result_article.InlineQueryResultArticle attribute), 233 url (aiogram.types.link_preview_options.LinkPreviewOptions utMember attribute), 151 url (aiogram.types.login_url.LoginUrl attribute), 153 riutChatMem(biogram.types.message_entity.MessageEntity attribute), 206 enable(Biogram.types.user.User property), 221 url (aiogram.types.web_app_info.WebAppInfo attribute), MemberRestricted url (aiogram.types.webhook_info.WebhookInfo attribute), 300 url (aiogram.utils.callback_answer.CallbackAnswer property), 580 Url (class in aiogram.utils.formatting), 584 URLInputFile (class in aiogram.types.input_file), 139, 488 USD (aiogram.enums.currency.Currency attribute), 477 use_independent_chat_permissions (aiogram.methods.restrict_chat_member.RestrictChatMember attribute), 372 use_independent_chat_permissions (aiogram.methods.set_chat_permissions.SetChatPermissions attribute), 415

```
attribute), 30
                                                                                                 attribute), 362
user (aiogram.types.chat_boost_source_gift_code.ChatBoostsenraeCiftCoden.methods.promote_chat_member.PromoteChatMember
             attribute), 54
                                                                                                 attribute), 368
user (aiogram.types.chat_boost_source_giveaway.ChatBooxtSenradQivegwayn.methods.replace_sticker_in_set.ReplaceStickerInSet
              attribute), 55
                                                                                                 attribute), 309
user (aiogram.types.chat boost source premium.ChatBoostSearced (aiogram.methods.restrict chat member.RestrictChatMember
             attribute), 55
                                                                                                 attribute), 372
user (aiogram.types.chat_member_administrator.ChatMembseAdministrator.methods.set_chat_administrator_custom_title.SetChatAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdministrator.chatMembseAdmin
             attribute), 100
                                                                                                 attribute), 412
user (aiogram.types.chat_member_banned.ChatMemberBansed_id (aiogram.methods.set_game_score.SetGameScore
             attribute), 102
                                                                                                 attribute), 454
user (aiogram.types.chat_member_left.ChatMemberLeft user_id (aiogram.methods.set_passport_data_errors.SetPassportDataErro
             attribute), 102
                                                                                                 attribute), 470
user (aiogram.types.chat_member_member.ChatMemberMwsbr_id (aiogram.methods.set_sticker_set_thumbnail.SetStickerSetThumbr
             attribute), 103
                                                                                                 attribute), 317
user (aiogram.types.chat_member_owner.ChatMemberOwnwer_id (aiogram.methods.unban_chat_member.UnbanChatMember
             attribute), 103
                                                                                                 attribute), 427
user (aiogram.types.chat_member_restricted.ChatMemberRestricted(aiogram.methods.upload_sticker_file.UploadStickerFile
             attribute), 104
                                                                                                 attribute), 320
user (aiogram.types.game_high_score.GameHighScore user_id (aiogram.types.bot_command_scope_chat_member.BotCommand_
             attribute), 302
                                                                                                 attribute), 29
user (aiogram.types.message_entity.MessageEntity at-
                                                                                  user_id (aiogram.types.contact.Contact attribute), 127
                                                                                   user_id (aiogram.types.shared_user.SharedUser at-
             tribute), 206
user (aiogram.types.message reaction updated.MessageReactionUpdatibalte), 218
                                                                                   user_id (aiogram.types.user_shared.UserShared at-
             attribute), 210
user (aiogram.types.poll_answer.PollAnswer attribute),
                                                                                                 tribute), 223
              213
                                                                                   user_ids (aiogram.types.users_shared.UsersShared at-
user (aiogram.utils.web_app.WebAppInitData attribute),
                                                                                                 tribute), 223
              574
                                                                                   user_is_bot(aiogram.types.keyboard_button_request_user.KeyboardButt
User (class in aiogram.types.user), 220
                                                                                                 attribute), 149
user_administrator_rights
                                                                                   user_is_bot(aiogram.types.keyboard_button_request_users.KeyboardBu
             (aiogram.types.keyboard_button_request_chat.KeyboardButtotrRaque)stChat
             attribute), 149
                                                                                   user_is_premium(aiogram.types.keyboard_button_request_user.Keyboar
user_chat_id(aiogram.types.business_connection.BusinessConnectionribute), 150
             attribute), 30
                                                                                   user_is_premium(aiogram.types.keyboard_button_request_users.Keyboa
user_chat_id(aiogram.types.chat_join_request.ChatJoinRequest attribute), 150
             attribute), 61
                                                                                   USER_SHARED (aiogram.enums.content_type.ContentType
user\_id(aiogram.methods.add\_sticker\_to\_set.AddStickerToSet
                                                                                                 attribute), 475
              attribute), 302
                                                                                   user_shared
                                                                                                           (aiogram.types.message.Message
user_id(aiogram.methods.approve_chat_join_request.ApproveChatJtoihRtec)u&sf3
                                                                                   UserChatBoosts
             attribute), 322
                                                                                                                                  (class
                                                                                                                                                                in
user_id(aiogram.methods.ban chat member.BanChatMember
                                                                                                 aiogram.types.user_chat_boosts), 222
                                                                                   username (aiogram.types.chat.Chat attribute), 35
              attribute), 323
user_id (aiogram.methods.create_new_sticker_set.CreateNusseStrindomeStatiogram.types.chat_full_info.ChatFullInfo at-
             attribute), 304
                                                                                                 tribute), 58
user_id (aiogram.methods.decline_chat_join_request.Decline@thundeinReigngrstm.types.chat_shared.ChatShared at-
             attribute), 335
                                                                                                 tribute), 126
user_id(aiogram.methods.get_chat_member.GetChatMembsername
                                                                                                           (aiogram.types.shared_user.SharedUser
              attribute), 351
                                                                                                 attribute), 218
user_id(aiogram.methods.get_game_high_scores.GetGamaHerhamereniogram.types.user.User attribute), 221
                                                                                   username
                                                                                                      (aiogram.utils.web_app.WebAppChat
              attribute), 452
                                                                                                                                                              at-
user_id(aiogram.methods.get_user_chat_boosts.GetUserChatBooststribute), 576
             attribute), 362
                                                                                   username
                                                                                                      (aiogram.utils.web app.WebAppUser
user_id(aiogram.methods.get user profile photos.GetUserProfilePhotibuste), 575
```

UserPro	ofilePhotos	(class	in	VIDEO_CHAT_ENDED (aiogram.enums.content_type.ContentType
		er_profile_photos), 222		attribute), 475
users	(aiogram.types.us	ers_shared.UsersShared	at-	video_chat_ended (aiogram.types.message.Message
	tribute), 223			attribute), 162
users(		_chat_participants_invite	d.Vid	ide <b>v CdeoPoha</b> t <u>panartageapl</u> ants_invited
	attribute), 226			(aiogram.enums.content_type.ContentType
USERS_		nums.content_type.Conten	tType	
	attribute), 474			<pre>video_chat_participants_invited</pre>
users_	shared ( <i>aiogram</i> tribute), 161	.types.message.Message	at-	(aiogram.types.message.Message attribute), 162
UserSh	ared (class in aiogi	ram.types.user_shared), 2	23	VIDEO_CHAT_SCHEDULED
UsersSl	nared (class in aiog	gram.types.users_shared),	223	(aiogram.enums.content_type.ContentType
UTILIT	Y_BILL (aiogram.er	nums.encrypted_passport_	_eleme	ment.Encryp <b>wettPlossip)</b> y <b>:t</b> Etement
	attribute), 478			video_chat_scheduled
UYU (aio	gram.enums.currer	cy.Currency attribute), 47	78	(aiogram.types.message.Message attribute),
UZS (aio	gram.enums.curren	cy.Currency attribute), 47	78	162
V				VIDEO_CHAT_STARTED (aiogram.enums.content_type.ContentType attribute), 475
value (	aiogram.tvpes.dice.	Dice attribute), 127		<pre>video_chat_started (aiogram.types.message.Message</pre>
		end_contact.SendContact	at-	attribute), 162
	tribute), 382			<pre>video_duration(aiogram.types.inline_query_result_video.InlineQueryResult_</pre>
vcard (	aiogram.tvpes.conte	act.Contact attribute), 127	,	attribute), 268
vcard(	aiogram.types.inlind attribute), 252	e_query_result_contact.In	lineQ	Queighesuficential (aiogram.types.inline_query_result_cached_video.InlineQ attribute), 249
vcard(	aiogram.types.input	contact message conten	ıt.Inpı	pu <b>rcidaachaigha</b> ge <b>c</b> iosseum.types.inline_query_result_video.InlineQueryResu
`	attribute), 271		•	attribute), 268
VENUE	(aiogram.enums.co	ontent_type.ContentType		attribute), 473
VENUE (a	aiogram.enums.inlii	ne query result type.Inlin	1еОие	uerYides04495e (aiogram.methods.send_video_note.SendVideoNote
	attribute), 479			attribute), 407
venue (	aiogram.types.exter	nal_reply_info.ExternalRe	eplyIn	Inflyideo_note(aiogram.types.external_reply_info.ExternalReplyInfo
	attribute), 130	- 1 - V		attribute), 130
venue (	aiogram.types.mess	age.Message attribute), 16	50	<pre>video_note (aiogram.types.message.Message attribute),</pre>
Venue (	class in aiogram.tyj	pes.venue), 224		159
via_bo	t (aiogram.types.me	essage.Message attribute),	159	video_url(aiogram.types.inline_query_result_video.InlineQueryResultVi
via_ch	at_folder_invit	e_link		attribute), 267
	(aiogram.types.ch attribute), 106	at_member_updated.Cha	tMem	mb <b>videaarii</b> dth(aiogram.types.inline_query_result_video.InlineQueryResul attribute), 268
via_jo:	in_request( <i>aiogr</i>	am.types.chat_member_u <sub>l</sub>	odated	red.Virlandensberogedated (class in
	attribute), 106			aiogram.types.video_chat_ended), 225
VIDEO	(aiogram.enums.co	ontent_type.ContentType	at-	VideoChatParticipantsInvited (class in
	tribute), 473			aiogram.types.video_chat_participants_invited),
VIDEO (	aiogram.enums.inlii	ne_query_result_type.Inlin	1eQue	ueryResultType6
	attribute), 479			VideoChatScheduled (class in
VIDEO (	aiogram.enums.inpi	ıt_media_type.InputMedia	<i>аТуре</i>	
	attribute), 480			VideoChatStarted (class in
VIDEO (		ker_format.StickerFormat	t at-	aiogram.types.video_chat_started), 226
	tribute), 483			VideoNote (class in aiogram.types.video_note), 227
video	tribute), 404	s.send_video.SendVideo	at-	attribute), 484
video(	aiogram.types.exter	nal_reply_info.ExternalRe	eplyIn	Inf ND (aiogram.enums.currency.Currency attribute), 478
	attribute), 130			VOICE (aiogram.enums.content_type.ContentType at-
		age.Message attribute), 15	59	tribute), 473
Video (	class in aiogram ty	nas vidao) 224		VOICE (aiogram.enums.inline query result type.InlineQueryResultType

```
attribute), 479
                                                                                                                                                                                                                          width (aiogram.types.animation.Animation attribute), 19
voice
                                     (aiogram.methods.send voice.SendVoice
                                                                                                                                                                                                                    width(aiogram.types.input media animation.InputMediaAnimation
                                     tribute), 409
                                                                                                                                                                                                                                                              attribute), 141
\verb"voice" (aiogram.types.external_reply_info. External ReplyInfwidth (aiogram.types.input\_media\_video.InputMediaVideo, InputMediaVideo, Input
                                     attribute), 130
                                                                                                                                                                                                                                                               attribute), 145
voice (aiogram.types.message.Message attribute), 160
                                                                                                                                                                                                                         width (aiogram.types.photo size.PhotoSize attribute),
Voice (class in aiogram.types.voice), 227
voice_duration(aiogram.types.inline query result voiceshidhtheQuiogramstypes.inline query result voiceshidhtequiogramstypes.inline q
                                     attribute), 270
                                                                                                                                                                                                                         width (aiogram.types.video.Video attribute), 225
voice_file_id(aiogram.types.inline_query_result_cachedimniex.IndionarQue(nxiRgsult)QuphsedVoiaway.Giveaway at-
                                    attribute), 251
                                                                                                                                                                                                                                                              tribute), 135
voice_url (aiogram.types.inline_query_result_voice.Inline@naneResoluWti(@iogram.types.giveaway_completed.GiveawayCompleted
                                    attribute), 269
                                                                                                                                                                                                                                                              attribute), 135
voter_chat (aiogram.types.poll_answer.PollAnswer at-
                                                                                                                                                                                                                         winner_count (aiogram.types.giveaway_winners.GiveawayWinners
                                    tribute), 212
                                                                                                                                                                                                                                                               attribute), 136
voter_count (aiogram.types.poll_option.PollOption at-
                                                                                                                                                                                                                         winners (aiogram.types.giveaway_winners.GiveawayWinners
                                    tribute), 213
                                                                                                                                                                                                                                                              attribute), 136
                                                                                                                                                                                                                          winners_selection_date
W
                                                                                                                                                                                                                                                              (aiogram.types.giveaway.Giveaway attribute),
was_refunded(aiogram.types.giveaway_winners.GiveawayWinners 135
                                                                                                                                                                                                                          winners_selection_date
                                    attribute), 137
{\tt watcher} \ (a iogram. types. proximity\_alert\_triggered. ProximityAlertTriggered. Triggered. ProximityAlertTriggered. Triggered. Triggered.
                                                                                                                                                                                                                                                               attribute), 136
                                     attribute), 213
\verb|WEB\_APP| (a iogram.enums.menu\_button\_type.MenuButtonType = \verb|local_file| (a iogram.client.telegram.TelegramAPIServer) | the properties of the properties
                                                                                                                                                                                                                                                               attribute), 14
                                     attribute), 481
web_app (aiogram.types.inline_keyboard_button.InlineKeybbardBuffGESS_ALLOWED
                                                                                                                                                                                                                                                              (aiogram.enums.content_type.ContentType
                                    attribute), 138
web_app (aiogram.types.inline_query_results_button.InlineQueryResultsButt6h 474
                                                                                                                                                                                                                          write_access_allowed
                                     attribute), 270
                                                                                                                                                                                                                                                               (aiogram.types.message.Message
                                                                                                                                                                                                                                                                                                                                                                                                      attribute),
web_app (aiogram.types.keyboard_button.KeyboardButton
                                                                                                                                                                                                                                                               161
                                    attribute), 147
                                                                                                                                                                                                                         WriteAccessAllowed
                                                                                                                                                                                                                                                                                                                                                               (class
                                                                                                                                                                                                                                                                                                                                                                                                                                   in
web_app (aiogram.types.menu_button.MenuButton at-
                                                                                                                                                                                                                                                               aiogram.types.write_access_allowed), 229
                                     tribute), 154
\verb|web_app| (a iogram.types.menu\_button\_web\_app.MenuButtonWebApp|)|
                                    attribute), 155
WEB_APP_DATA (aiogram.enums.content_type.ContentType x_shift (aiogram.types.mask_position.MaskPosition at-
                                    attribute), 475
                                                                                                                                                                                                                                                               tribute), 278
web_app_data (aiogram.types.message.Message
                                    tribute), 162
web\_app\_name~(aiogram.types.write\_access\_allowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4llowed.WriteAccess\_4ll
                                    attribute), 229
                                                                                                                                                                                                                                                              tribute), 278
attribute), 450
                                                                                                                                                                                                                         YELLOW (aiogram.enums.topic icon color.TopicIconColor
WebAppChat (class in aiogram.utils.web_app), 576
                                                                                                                                                                                                                                                               attribute), 484
WebAppData (class in aiogram.types.web app data), 228
                                                                                                                                                                                                                         YER (aiogram.enums.currency.Currency attribute), 478
WebAppInfo (class in aiogram.types.web_app_info), 228
                                                                                                                                                                                                                         Ζ
WebAppInitData (class in aiogram.utils.web_app), 574
WebAppUser (class in aiogram.utils.web_app), 575
                                                                                                                                                                                                                         ZAR (aiogram.enums.currency.Currency attribute), 478
WebhookInfo (class in aiogram.types.webhook_info),
width(aiogram.methods.send_animation.SendAnimation
                                    attribute), 375
width
                                     (aiogram.methods.send video.SendVideo
                                    tribute), 404
```